Recovery Amendment to E-NNI 2.0 – RSVP-TE Signaling

OIF-ENNI-RSVP-02.2

February 4, 2014

Implementation Agreement created and approved
by the Optical Internetworking Forum
www.oiforum.com

The OIF is an international non profit organization with over 100 member companies, including the world's leading carriers and vendors. Being an industry group uniting representatives of the data and optical worlds, OIF's purpose is to accelerate the deployment of interoperable, cost-effective and robust optical internetworks and their associated technologies. Optical internetworks are data networks composed of routers and data switches interconnected by optical networking elements.

With the goal of promoting worldwide compatibility of optical internetworking products, the OIF actively supports and extends the work of national and international standards bodies. Working relationships or formal liaisons have been established with CFP-MSA, COAST, Ethernet Alliance, Fibre Channel T11, IEEE 802.1, IEEE 802.3, IETF, InfiniBand, ITU-T SG13, ITU-T SG15, MEF, ONF, Rapid I/O, SAS T10, SFF Committee, TMF and TMOC.

For additional information contact:
The Optical Internetworking Forum, 48377 Fremont Blvd.,
Suite 117, Fremont, CA 94538
510-492-4040 ☞ info@oiforum.com

www.oiforum.com

Working Group: **Networking and Operations**

TITLE: **Recovery Amendment to E-NNI 2.0 Signaling – RSVP-TE Signaling**

SOURCE:

SOURCE:

TECHNICAL EDITOR

Remi Theillaud
Marben Products
176 rue Jean Jaures
92800 Puteaux
France
Phone: +33 (1) 79 62 10 22
Email: remi.theillaud@marben-products.com

WORKING GROUP CHAIR

Evelyne Roch
Huawei Technologies
303 Terry Fox Drive Suite 400
Ottawa, ON K2K 3J1
Canada
Phone: +1 613 595 1900 x1612
Email:evelyne.roch@huawei.com

1 Table of Contents

2 List of Figures

3 List of Tables

4 Introduction

The scope of this amendment is to define the E-NNI RSVP-TE Signaling Interface based on the E-NNI 2.0 Signaling [OIF-ENNI2.0-SIG], Multilayer Amendment [OIF-ENNI-ML-AM-01.0] and [OIF-ENNI-RSVP-02.1], and Recovery Amendment [OIF-ENNI-REC-AM-01.0]. All extensions for this amendment are in green colored font to help the reader identify the changes (all extensions for the multilayer amendment are in blue colored). A summary of all changes can also be found in section0.

5 Terminology and Abbreviations

ASON	Automatically Switched Optical Network (see [G.8080])
BwAR	Bridge with Automatic Roll (see [OIF-ENNI-REC-AM-01.0])
B&R	Bridge & Roll (see [OIF-ENNI-REC-AM-01.0])
CBS	Committed Burst Size
CE-VLAN ID	Customer Edge Virtual Local Area Network Identifier
CIR	Committed Information Rate
DCSC	Data Channel Switching Capable
DEN	Domain Egress Node (see [OIF-ENNI-REC-AM-01.0])
DIN	Domain Ingress Node (see [OIF-ENNI-REC-AM-01.0])
EBS	Excess Burst Size
EIR	Excess Information Rate
E-NNI	External NNI (see [G.8080])
eNNI-D	The logical control plane entity that terminates E-NNI signaling in the downstream direction with respect to control plane initiation
eNNI-U	The logical control plane entity that terminates E-NNI signaling in the upstream direction with respect to control plane initiation
EPL	Ethernet Private Line
ERO	Explicit Route Object
EVC	Ethernet Virtual Connection
EVPL	Ethernet Virtual Private Line
GMPLS	Generalized MPLS
IETF	Internet Engineering Task Force
I-NNI	Internal NNI (see [G.8080])
IP	Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6)
IPsec	Internet Protocol Security (see [OIF-SEC] and [SecAdd])
LCAS	Link Capacity Adjustment Scheme (see [G.7042])
LIH	Logical Interface Handle
LSP	Label Switched Path
LSR	Label Switched Router
MT	Multiplier
NCC	Network Call Controller
NNI	Network Node Interface

Node ID	Node Identifier (see [G.7715.1])[1]
OSPF-TE	Open Shortest Path First (OSPF) Traffic Engineering
OTN	Optical Transport Network (see [G.709])
PC	Protocol Controller (see [G.8080])
RA	Routing Area (see [G.8080])
RC	Routing Controller
RRO	Record Route Object
RSVP	Resource Reservation Protocol (see [RFC2205])
RSVP-TE	RSVP Traffic Engineering (see [RFC3209])
SC	Switched Connection service (see [G.8080])
SC PC ID	Signaling Controller Protocol Controller Identifier
SCN	Signaling Communications Network (see [G.7712])
SDH	Synchronous Digital Hierarchy (see [G.707])
SE	Shared Explicit
SMI	Structure of Management Information (see [RFC5612])
SNP	Subnetwork Point (see [G.8080])
SNPP	Subnetwork Point Pool (see [G.8080])
SONET	Synchronous Optical Network (see [T1.105])
SPC	Soft Permanent Connection service (see [G.8080])
TCP	Termination Connection Point (see [G.805])
TDM	Time Division Multiplexing
TLV	Type-Length-Value encoding
TNA	Transport Network Assigned (see [G.8080])
UNI	User Network Interface (see [OIF-UNI-02.0], [G.8080])
UNI-C	The logical entity that performs UNI signaling on the user device side.
UNI-N	The logical entity that performs UNI signaling on the network device side.
VCAT	Virtual Concatenation

6 RSVP-TE Extensions for E-NNI Signaling

The RSVP-TE extensions to support the E-NNI signaling mechanism are based on the protocol capabilities as specified in [RFC2205], [RFC2961], [RFC3209], [RFC3473], [RFC3474], [RFC3476], [RFC4328], and [RFC4606]. This section provides a summary of the messages, objects, and error codes relevant to this Implementation Agreement.

6.1 Overview of RSVP-TE Operation

An overview of the basic RSVP-TE operation may be found in [RFC3209] and [RFC3473]. When a eNNI-U (eNNI-D) sends an RSVP message, it MUST address the message directly to its eNNI-D (eNNI-U) peer. The peer's SCN address is used for this purpose. A node (signaling

[1] The Node ID identifies a node in the transport topology graph. This definition differs from that given in the OIF UNI 2.0 specification.

protocol controller) should use IP encapsulation of RSVP messages. Furthermore, the IPv4 Router Alert option MUST NOT be set in any RSVP messages. The IPv4 header fields are shown in Table 1.

Table 1: IPv4 Header for E-NNI RSVP Messages

IPv4 Header Values for E-NNI RSVP Messages	
Version	4
Header Length	5
TOS	As defined in [RFC791]
Total Length	Message length
Identification	As defined in [RFC791]
Flags	As defined in [RFC791]
Fragment Offset	As defined in [RFC791]
TTL	>= 1
Protocol	46
Header Checksum	As defined in [RFC791]
Source Address	eNNI-U/eNNI-D SC PC SCN IP address
Destination Address	eNNI-D/eNNI-U SC PC SCN IP address

The format of the RSVP <Common Header> object is defined in [RFC2205], Section 3.1.1.

The flag field of the RSVP common header MUST be set to 1, to indicate support of the Bundle and Srefresh messages. As a result, an implementation MUST be able to process Bundle and Srefresh messages received from a neighbor, and MAY choose to use bundling and Srefresh when sending messages.

6.2 Messages and Error Codes

Table 2 provides a mapping of the abstract connection messages to specific RSVP-TE messages used to support signaling across the E-NNI interface.

Table 2: Mapping of Abstract Messages to RSVP-TE Messages

Abstract Messages	**RSVP-TE Messages**
ConnectionSetupRequest	Path
ConnectionSetupIndication	Resv In case of error - PathErr
ConnectionSetupConfirm	ResvConf In case of error - PathTear
ConnectionReleaseRequest	Path or Resv (with Delete and Reflect bits (D&R bits))

	Path or Resv (with Admin and Reflect (A&R bits)) – only for compatibility with UNI/E-NNI 1.0
ConnectionReleaseIndication	PathTear or PathErr (w/ Path_State_Removed flag)
ConnectionQueryRequest	Implicit
ConnectionQueryIndication	Implicit
ConnectionNotification	Notify (w/ D bit set) or PathErr
ConnectionModifyRequest	Path
ConnectionModifyIndication	Resv In case of error - PathErr
ConnectionModifyConfirm	ResvConf In case of error - PathTear
Signaling Adjacency Maintenance	Hello
ConnectionActivationRequest	Path
ConnectionActivationIndication	Resv
MakePersistentRequest	Path
MakePersistentIndication	Resv

Table 3 provides information on which abstract messages are associated with each RSVP message.

Table 3: RSVP Messages by Abstract Message

RSVP-TE Messages	Abstract Messages
Path	ConnectionSetupRequest ConnectionReleaseRequest ConnectionModifyRequest ConnectionActivationRequest MakePersistentRequest
Resv	ConnectionSetupIndication ConnectionReleaseRequest ConnectionModifyIndication ConnectionActivationIndication MakePersistentIndication

RSVP-TE Messages	Abstract Messages
PathErr	ConnectionSetupIndication ConnectionReleaseIndication ConnectionNotification ConnectionModifyIndication
ResvConf	ConnectionSetupConfirm ConnectionModifyConfirm
PathTear	ConnectionSetupConfirm ConnectionReleaseIndication ConnectionModifyConfirm
Notify (w/ D bit set)	ConnectionNotification
Hello	Signaling Adjacency Maintenance

Table 4 provides a mapping of the abstract error codes to specific RSVP-TE error codes and values used to support signaling across the E-NNI interface. These error codes are specified in [RFC2205], [RFC3209], and [RFC4974].

Table 4: Mapping of Abstract Error Codes to RSVP-TE Error Codes and Values

Abstract Errors	RSVP-TE Error Codes/Error Values
Calling Party Busy	ERROR_SPEC 24/5
Called Party Busy	ERROR_SPEC 24/103
Unauthorized sender (policy error)	ERROR_SPEC 2/100
Unauthorized receiver (policy error)	ERROR_SPEC 2/101
Invalid / unknown connection ID	ERROR_SPEC 24/102
Invalid / unknown call ID	ERROR_SPEC 24/105
Invalid SNP	ERROR_SPEC 24/6 or 24/11 or 24/12 or 24/14
Unavailable SNP	ERROR_SPEC 24/6 or 24/11 or 24/12 or 24/14
Invalid SNPP	ERROR_SPEC 24/104
Unavailable SNPP	ERROR_SPEC 24/104
Unavailable directionality	ERROR_SPEC 24/6 or 24/11
Invalid SPC SNP	ERROR_SPEC 24/106
Invalid route	ERROR_SPEC 24/1, 24/2, 24/3, or 24/7
Invalid recovery	ERROR_SPEC 24/15 or 24/100
Unavailable recovery	ERROR_SPEC 24/15 or 24/100
Unavailable service level	ERROR_SPEC 24/101 or 2/{any}
Service-affecting defect	ERROR_SPEC 24/5, 24/9, 24/100, 24/101 or 24/103
Non-service-affecting defect	General protocol error: general RSVP-TE error codes/values

6.2.1 Hello Message (Msg Type = 20 [RFC3209])

This message is specified in [RFC3209].

The Hello message is used to establish a signaling adjacency and for communications failure detection. It has the following format:

<Hello message> ::= <Common Header>

<HELLO>

<RESTART_CAP>

Hello messages are retransmitted periodically to an adjacent E-NNI signaling peer. The retransmission interval SHALL be administratively configurable. The default value is 5 seconds.

6.2.2 Path Message (Msg Type = 1 [RFC2205])

This message is specified in [RFC2205], with further extensions made by [RFC2961], [RFC3209], and [RFC3473].

<Path Message> ::=

<Common Header>

[[<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ...]

<MESSAGE_ID>

< SESSION> < RSVP_HOP>

<TIME_VALUES>
<GENERALIZED_LABEL_REQUEST>
<CALL_ID>
[<LABEL_SET> ...]
[<SESSION_ATTRIBUTE>]
[<EXPLICIT_ROUTE>]
<NOTIFY_REQUEST>
[<ADMIN_STATUS>]
<Generalized UNI>
[<OIF_PATH_VENDOR_PRIVATE_EXTENSION_TYPE_1>]
[<OIF_PATH_VENDOR_PRIVATE_EXTENSION_TYPE_3>]
[<POLICY_DATA> ...]
<sender descriptor>

Note: The GENERALIZED_UNI object is always present because it will either be provided by a UNI client initiating, adding, or deleting, a connection, or it will be provided by the ingress domain when initiating an SPC or hybrid connection to carry the source and destination TNAs, as well as other optional GENERALIZED_UNI sub-objects. This does not imply that intermediate NNI nodes should be able to process the GENERALIZED_UNI object, only that they are able to forward it between the source and destination of the connection. The format of the < Generalized UNI > and <sender descriptor> objects are described in [OIF-UNI-02.0-R2-RSVP], Section 9.1.3 "Path Message".

<sender descriptor> ::=
<SENDER_TEMPLATE>
<sender_tspec >
[<RECORD_ROUTE>]
[<SUGGESTED_LABEL>]
[<RECOVERY_LABEL>]
[<UPSTREAM_LABEL>]

<OIF_PATH_VENDOR_PRIVATE_EXTENSION_TYPE_1> ::=
[<OIF_INV_MUX_IF_ID>]
[<OIF_INV_MUX_SENDER_TSPEC>]
[<OIF_RECOVERY_STACK>]

<OIF_PATH_VENDOR_PRIVATE_EXTENSION_TYPE_3> ::=
[<OIF_LSP_TUNNEL_INTERFACE_ID>]

[<OIF_VENDOR_PRIVATE_ERO>]

[<OIF_VENDOR_PRIVATE_RRO>]

<sender_tspec>::=

<SONET_SDH_TSPEC> | <G709_TSPEC> | <ETH_TSPEC> | <INTSERV_TSPEC>

<OIF_RECOVERY_STACK> ::=

<RECOVERY_LIST>

<RECOVERY_LIST>:: =

<RECOVERY> | <RECOVERY_LIST> <RECOVERY>

<RECOVERY> ::=

<PROTECTION> [<ASSOCIATION>] [<PRIMARY_PATH_ROUTE>]

A Path message establishing a unidirectional connection over an NNI interface does not include an UPSTREAM_LABEL object.

6.2.3 Resv Message (Msg Type = 2 [RFC2205])

This message is specified in [RFC2205], with further extensions made by [RFC2961], [RFC3209], and [RFC3473]. The Resv message is used for connection creation and call/connection modification. The RESV_CONFIRM object may be included in the Resv message to request a ResvConf message to confirm the connection setup. Once the RESV_CONFIRM object has been included in the Resv message, it would normally be included in full refreshes of the Resv without generating additional ResvConf messages in response. A subsequent trigger change in the Resv message MAY result in a new ResvConf response. To force a new ResvConf message, the RESV_CONFIRM object SHOULD be removed from the Resv message and then inserted into a subsequent Resv message so that the change acts as a trigger for a new ResvConf.

The format of the E-NNI Resv message is as shown below:

<Resv Message> ::= <Common Header>

[[<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ...]

<MESSAGE_ID>

<SESSION> < RSVP_HOP >
<TIME_VALUES>

<CALL_ID>

<NOTIFY_REQUEST>

[<ADMIN_STATUS>]

[<POLICY_DATA> ...]
[<RESV_CONFIRM>]
[<OIF_RESV_VENDOR_PRIVATE_EXTENSION_TYPE_1>]
[<OIF_RESV_VENDOR_PRIVATE_EXTENSION_TYPE_3>]
<STYLE>
<FF flow descriptor> | <SE flow descriptor>

<FF flow descriptor> ::=
<SONET/SDH_FLOWSPEC> | <G.709 FLOWSPEC> | <ETH FLOWSPEC> |
<INTSERV FLOWSPEC>
<FILTER_SPEC>
<GENERALIZED_LABEL>
[<RECORD_ROUTE>]

<SE flow descriptor> ::=
<SONET/SDH_FLOWSPEC> | <G.709 FLOWSPEC> | <ETH FLOWSPEC> |
<INTSERV FLOWSPEC>
<FILTER_SPEC>
<GENERALIZED_LABEL>
[<RECORD_ROUTE>]

<OIF_RESV_VENDOR_PRIVATE_EXTENSION_TYPE_1> ::=
[<OIF_INV_MUX_IF_ID>]
[<OIF_INV_MUX_FLOWSPEC>]
[<OIF_RECOVERY_STACK>]

<OIF_RESV_VENDOR_PRIVATE_EXTENSION_TYPE_3> ::=
[<OIF_LSP_TUNNEL_INTERFACE_ID>]
[<OIF_VENDOR_PRIVATE_RRO>]
<OIF_RECOVERY_STACK> ::=
<RECOVERY_LIST>

<RECOVERY_LIST>:: =
<RECOVERY> | <RECOVERY_LIST> <RECOVERY>

<RECOVERY> ::= <PROTECTION>

6.2.4 ResvConf Message (Msg Type = 7 [RFC2205])

This message is specified in [RFC2205], with further extensions made by [RFC2961], [RFC3209], and [RFC3473]. Note that the Call Name (CALL_ID) is not included in the ResvConf message, in accordance with [RFC3474]. As a result, the attributes are not aligned with the abstract Connection Setup Confirm message. It is still possible to correlate the ResvConf message with the proper Resv state based on the SESSION and flow descriptor.

The ResvConf message is originated at the source UNI-C to acknowledge the receipt of a trigger[2] Resv message that includes a RESV_CONFIRM Object. ResvConf messages are sent from the source UNI-C to the corresponding UNI-N, and from the destination UNI-N to the destination UNI-C. While the E-NNI processes ResvConf on a hop by hop basis, the message scope is end-to-end, and the network MUST relay the ResvConf message from source UNI-N to destination UNI-N.

The format of the ResvConf message is shown below:

<ResvConf message> ::= <Common Header>
[[<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ...]
<MESSAGE_ID>
<SESSION> <ERROR_SPEC>
<RESV_CONFIRM>
<STYLE>
<FLOW_SPEC > | <FILTER_SPEC>

6.2.5 PathTear Message (Msg Type = 5 [RFC2205])

This message is specified in [RFC2205], with further extensions made by [RFC2961], [RFC3209], and [RFC3473].

<PathTear Message> ::= <Common Header>
[[<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ...]
<MESSAGE_ID>
<SESSION>
<CALL_ID>
<RSVP_HOP>
<sender descriptor> /* (see Section 6.2.2) */

[2] A trigger Resv message is a message that modifies the reservation state. Examples include the original Resv message sent during connection establishment in response to the first Path message and the Resv message that includes a change in the FILTER_SPEC object sent during connection modification.

6.2.6 PathErr Message (Msg Type = 3 [RFC2205])

This message is specified in [RFC2205], with further extensions made by [RFC2961], [RFC3209], and [RFC3473]. The PathErr message is used to report errors and for connection deletion.

The format of the E-NNI PathErr message is as follows:

<PathErr message> ::= <Common Header>
[[<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ...]
<MESSAGE_ID>
<SESSION>
<CALL_ID>
<ERROR_SPEC>
[<ACCEPTABLE_LABEL_SET>]
[<POLICY_DATA> …]
<sender descriptor> /* see Section 6.2.2*/

6.2.7 Notify Message (Msg Type = 21 [RFC3473])

The Notify message is used to support intermediate node initiated deletion, and the ADMIN_STATUS object is mandatory. This message is specified by [RFC3473].

<Notify message> ::= <Common Header>
[[<MESSAGE_ID_ACK> | <MESSAGE_ID_NACK>] ...]
<MESSAGE_ID>
<ERROR_SPEC>
<notify session lists> /* 1 or more notify sessions */

<notify session lists> ::= [<notify session lists>]
<upstream notify session> | <downstream notify session>
<upstream notify session> ::= <SESSION> <CALL_ID> <ADMIN_STATUS[3]>
[<POLICY_DATA>...] <sender descriptor>
<downstream notify session> ::= <SESSION> <CALL_ID> <ADMIN_STATUS>
[<POLICY_DATA>...] <flow descriptor list>
<flow descriptor list> ::=

[3]In E-NNI Signaling 2.0, the ADMIN_STATUS is mandatory in the Notify message. This is because the Notify message is used in E-NNI Signaling 2.0 to support intermediate node initiated deletion.

<FF flow descriptor> | <SE flow descriptor> /* 1 or more flow descriptor */

Note that the downstream notify session adds the ADMIN_STATUS object as per [RFC4974]. It enables the following situations to be handled:

a) a UNI/ENNI 1.0 node allows sending network deletion both upstream and downstream. If there was a UNI/ENNI 1.0 node upstream of an ENNI 2.0 interface, the ENNI 2.0 interface may receive a downstream network deletion request from the UNI/ENI 1.0 node.

b) a network I-NNI domain may initiate a downstream graceful deletion.

Note that the IETF model allows a NOTIFY Message to be sent to any recipient. The E-NNI 2.0 IA uses the NOTIFY Message only to perform Graceful Deletions for intermediate nodes, so the NOTIFY Message can only be directed to nodes with an upstream or downstream signaling adjacency with this node. The NOTIFY message has end-to-end significance, though it is processed on a hop by hop basis, and MUST be forwarded to continue the intermediate deletion message flow.

6.2.8 Srefresh Message

This message is specified by [RFC2961].

6.2.9 Ack Message

This message is specified by [RFC2961].

6.3 Attributes

Table 5 provides a mapping of the abstract attributes to specific RSVP-TE objects used to support signaling across the E-NNI interface.

Table 5: Mapping of Abstract Attributes to RSVP-TE Objects

Abstract Attributes	**RSVP-TE Objects**
Source TNA name	GENERALIZED_UNI/Source_TNA
Destination TNA name	GENERALIZED_UNI/Destination_TNA
DEST SNP ID	GENERALIZED_UNI/SPC_LABEL
Initiating NCC PC ID	SENDER_TEMPLATE
Terminating NCC PC ID	SESSION
Connection name	SESSION + SENDER_TEMPLATE
Call name	CALL_ID
SNP ID	SENDER_TSPEC, RSVP_HOP, LABEL, GENERALIZED_UNI/EGRESS_LABEL
SNPP ID	RSVP_HOP[4], LABEL_SET

[4] The “IPv4 Next/Previous Hop Address” field of the RSVP_HOP object is not part of the SNPP ID as it is a control address.

Abstract Attributes	**RSVP-TE Objects**
	OIF_INV_MUX_IF_ID
Directionality	Implied by UPSTREAM_LABEL
Explicit route	EXPLICIT_ROUTE, RECORD_ROUTE, OIF_VENDOR_PRIVATE_ERO, OIF_VENDOR_PRIVATE_RRO
Service level	GENERALIZED_UNI/DIVERSITY, GENERALIZED_UNI/SERVICE_LEVEL, POLICY_DATA, SESSION_ATTRIBUTE
Contract ID	POLICY_DATA
Encoding Type	GENERALIZED_LABEL_REQUEST/ LSP_ENC_TYPE
Switching Type	GENERALIZED_LABEL_REQUEST/ SWITCHING_TYPE
SONET/SDH	SONET/SDH_TSPEC, SONET/SDH_FLOWSPEC
VCAT	OIF_INV_MUX _TSPEC, OIF_INV_MUX_FLOWSPEC, INTSERV_TSPEC, INTSERV_FLOWSPEC
OTN	G709_TSPEC, G709_FLOWSPEC
Ethernet traffic parameters	ETHERNET_TSPEC, ETHERNET_FLOWSPEC
Generalized Payload Identifier	GENERALIZED_LABEL_REQUEST/G-PID
Connection Status	ADMIN_STATUS
Multilayer Signaling Based Discovery	OIF_LSP_TUNNEL_INTERFACE_ID
Recovery	OIF_RECOVERY_STACK ENNI Recovery sub-object: ASSOCIATION ENNI Recovery sub-object: PROTECTION ENNI Recovery sub-object: PRIMARY_PATH_ROUTE

Table 6 provides a summary of the various objects used to support E-NNI signaling, along with codepoints assigned to the objects that are relevant to support the applications across the E-NNI interface. Note that this table only specifies the codepoints that are relevant to the OIF E-NNI specification and does not list all available codepoints (e.g., SENDER_TEMPLATE only lists C-type 7). Also, unless otherwise specified in this section, formats of these objects are as defined in the associated reference.

Table 6: Summary of RSVP-TE E-NNI Objects

RSVP-TE Object	Class-Num/ C-type[/ Type/ [Sub-type]]	Reference
ACCEPTABLE_LABEL_SET	130/{same as label_set}	[RFC3473], [RFC4328] Refer to section 6.3.13
ADMIN_STATUS[1]	196/1	[RFC3473]
CALL_ID	230/<1,2>	[RFC3474]
ERROR_SPEC	6/3/{same as RSVP_HOP}[2]	[RFC2205], [RFC3209], [RFC3471], [RFC3473] Refer to Section 6.3.1
EXPLICIT_ROUTE	20/1/<3,4>[3]	[RFC3209], [RFC3473], [RFC3477] Refer to Section 6.3.2
FILTER_SPEC	10/{same as SENDER_TEMPLATE}	[RFC2205], [RFC3209], [RFC3473]
SONET/SDH_FLOWSPEC	9/4	[RFC4606]
G.709 FLOWSPEC	9/5	[RFC4328]
ETHERNET FLOWSPEC	9/6	[RFC6003]
INTSERV FLOWSPEC	9/2	[RFC2210] Refer to Section 6.3.10
GENERALIZED_UNI /DESTINATION_TNA	229/1/2/<1,2,3>	[OIF-UNI-02.0] Refer to Section 6.3.4
GENERALIZED_UNI /DIVERSITY	229/1/3/1	[OIF-UNI-02.0] Refer to Section 6.3.4
GENERALIZED_UNI /EGRESS_LABEL	229/1/4/1	[OIF-UNI-02.0] Refer to Section 6.3.4 Refer to section 6.3.13 for VCAT
GENERALIZED_UNI /SERVICE_LEVEL	229/1/5/1	[OIF-UNI-02.0] Refer to Section 6.3.4

RSVP-TE Object	Class-Num/ C-type[/ Type/ [Sub-type]]	Reference
GENERALIZED_UNI /SOURCE_TNA	229/1/1/<1,2,3>	[OIF-UNI-02.0] Refer to Section 6.3.4
GENERALIZED_UNI /SPC_LABEL[4]	229/4/2	[RFC3474] Refer to Section 6.3.4 Refer to section 6.3.13 for VCAT
HELLO_REQUEST/ HELLO_ACK	22/<1,2>	[RFC3209], [RFC3473]
RSVP_LABEL (GENERALIZED_LABEL)[5]	16/2	[RFC3473], [RFC4328] Refer to section 6.3.13 for VCAT
GENERALIZED _LABEL_REQUEST	19/4	[RFC3473], [RFC4328] Refer to section 6.3.13 for VCAT
LABEL_SET[6]	36/1	[RFC3473], [RFC4328] Refer to section 6.3.13 for VCAT
MESSAGE_ID	23/1	[RFC2961]
MESSAGE_ID_ACK/ MESSAGE_ID_NACK	24/<1,2>[7]	[RFC2961]
MESSAGE_ID_LIST	25/1	[RFC2961]
NOTIFY_REQUEST	195/1	[RFC3473]
POLICY_DATA	14/1	[RFC2205]
RECORD_ROUTE	21/{same as ERO}	[RFC3209], [RFC3473], [RFC3477], Refer to section 6.3.3
RECOVERY_LABEL	34/{same as RSVP_LABEL}	[RFC3473], [RFC4328] Refer to section 6.3.13 for VCAT
RESTART_CAP	131/1	[RFC3473]
RESV_CONFIRM	15/1	[RFC2205] Refer to Section 6.3.5

RSVP-TE Object	Class-Num/ C-type[/ Type/ [Sub-type]]	Reference
RSVP_HOP	3/3/<3,4,5>[8] 3/1	[RFC2205], [RFC3471], [RFC3473] Refer to Section 6.3.6
SENDER_TEMPLATE	11/7	[RFC2205], [RFC3209], [RFC3473] Refer to Section 6.3.7
SONET_SDH_TSPEC	12/4	[RFC4606]
G709_TSPEC	12/5	[RFC4328]
ETHERNET_TSPEC	12/6	[RFC6003]
INTSERV_TSPEC	12/2	[RFC2210] Refer to Section 6.3.10
SESSION	1/15	[RFC2205], [RFC3209], Refer to Section 6.3.8
SESSION_ATTRIBUTE	207/<1,7>	[RFC3209]
STYLE[9]	8/1	[RFC2205]
SUGGESTED_LABEL	129/{same as RSVP_LABEL}	[RFC3473] Refer to section 6.3.13 for VCAT
TIME_VALUES[10]	5/1	[RFC2205]
UPSTREAM_LABEL	35/{same as RSVP_LABEL}	[RFC3473], [RFC4328] Refer to section 6.3.13 for VCAT
OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_1	124/1 OIF_INV_MUX_IF_ID (1/1) OIF_INV_MUX_TSPEC (2/1) OIF_INV_MUX_FLOWSPEC (3/1) OIF_RECOVERY_STACK (4/1)	Refer to [OIF-RSVP-Ext] and section 6.3.11 Refer to section 6.3.11.3 for OIF_RECOVERY_STACK

RSVP-TE Object	Class-Num/ C-type[/ Type/ [Sub-type]]	Reference
OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3	252/1 OIF_LSP_TUNNEL_INTERFACE_ID (2/1) OIF_VENDOR_PRIVATE_ERO (3/1) OIF_VENDOR_PRIVATE_RRO (4/1) OIF_ML_ADAPTATION (5/1)	Refer to [OIF-RSVP-Ext] and section 6.3.12

Note 1: The absence of this object is equivalent to receiving an object containing values all set to zero (0).

Note 2: {text} where text is a comment.

Note 3: <…> indicates the different C-types or sub-types defined for the particular object.

Note 4: The port identifier contained in the SPC_LABEL sub-object is the logical port identifier assigned at the destination UNI-N; the port identifier contained in the EGRESS_LABEL sub-object is a logical port identifier assigned at the destination UNI-C.

Note 5: The format of LABEL is dependent on the signal types defined by LABEL_REQUEST object.

Note 6: A LABEL_SET object contains a list of "sub-channels" whose type is inferred from the label type field. Each of the sub-channels represents a label (wavelength, fiber, timeslot, etc.). A given LABEL_SET object MUST include a single label type. The interpretation of the label depends on the type of the link over which the label is to be used, so each sub-channel does NOT need its own header within the LABEL_SET object.

Note 7: MESSAGE_ID_NACK is a sub-type of MESSAGE_ID_ACK.

Note 8: RSVP_HOP Types 4 and 5 SHOULD NOT be generated but MUST be supported if received for E-NNI 1.0 backward compatibility.

Note 9: Both "fixed filter" and "shared explicit" styles are used.

Note 10: The value MUST be coordinated with Srefresh intervals to ensure proper refresh of the state information.

6.3.1 ERROR_SPEC

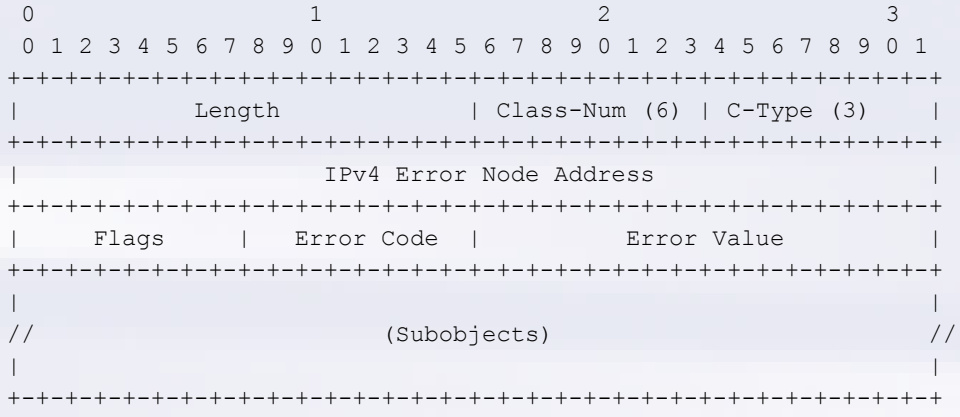
The IPv4 IF_ID_ERROR_SPEC (class = 6, C-Type =3) is defined in [RFC3473]. The IPv4 IF_ID_ERROR_SPEC MUST be supported. In E-NNI signaling, the error node address MUST be set to the SC PC ID of E-NNI (Identifier of eNNI-U or eNNI-D) that reported the error.

During graceful deletion, the path_State_Removed flag SHOULD be set, and Error Code 0 (confirmation) and Error Value 0 SHOULD be used.

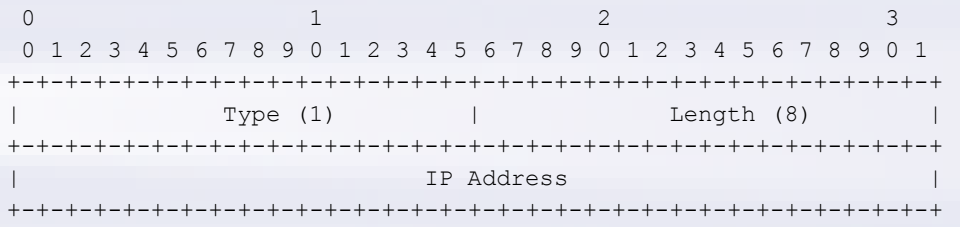
When the Error Code is non-zero, the IPv4 IF_ID_ERROR_SPEC SHOULD contain an IPv4 (Type 1) or an IPv4 IF_INDEX (Type 3) sub-object, as defined in [RFC3471]. The content of the IPv4 sub-object or IF_INDEX sub-object is significant for routing, and the identifiers used are those advertised by E-NNI 2.0 Routing and signaled in the ERO. The IF_INDEX sub-object corresponds to a hop in the ERO. In addition to either a Type 1 or Type 3 sub-object, further objects may be included using definitions in [RFC4920].

The Type 1 object reports a whole transport node failure, and the IPv4 Address is the Node ID of the failed node. The Type 3 sub-object also carries an interface ID, specifying a link failure at that transport node.

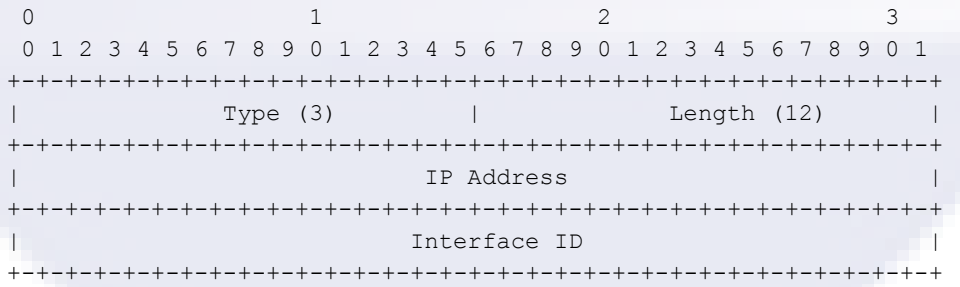
The format of the [RFC3473] IF_ID ERROR_SPEC, Type 1 sub-object is depicted below:



The format of the [RFC3473] IF_ID ERROR_SPEC, Type 1 sub-object is depicted below:

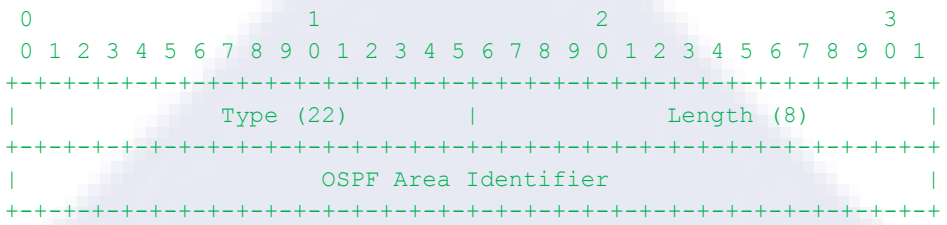


The format of the [RFC3473] IF_ID ERROR_SPEC, Type 3 sub-object is depicted below:



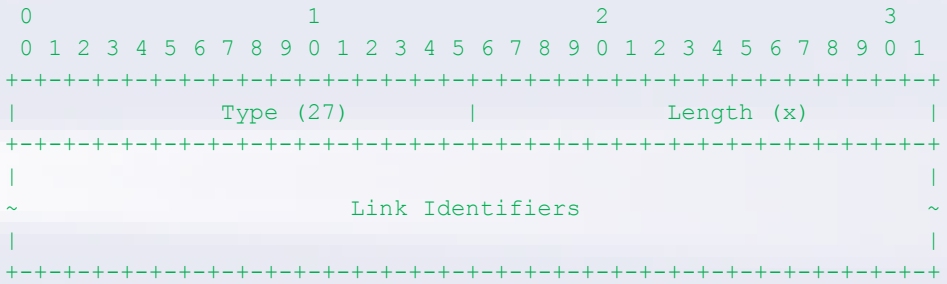
For failure notification during Recovery two additional sub-objects may be used: REPORTING_OSPF_AREA and LINK_EXCLUSIONS (see section 6.7.4).

The format of the [RFC4920] REPORTING_OSPF_AREA, Type 22 sub-object is depicted below:



Note that this sub-object is used to carry a recovery domain identifier (i.e. a Routing Area identifier), not an OSPF area ID.

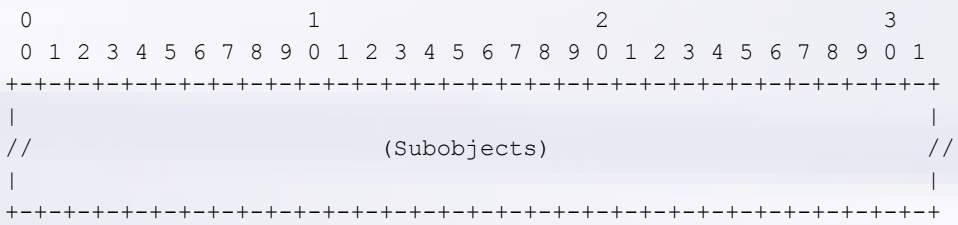
The format of the [RFC4920] LINK_EXCLUSIONS, Type 27 sub-object is depicted below:



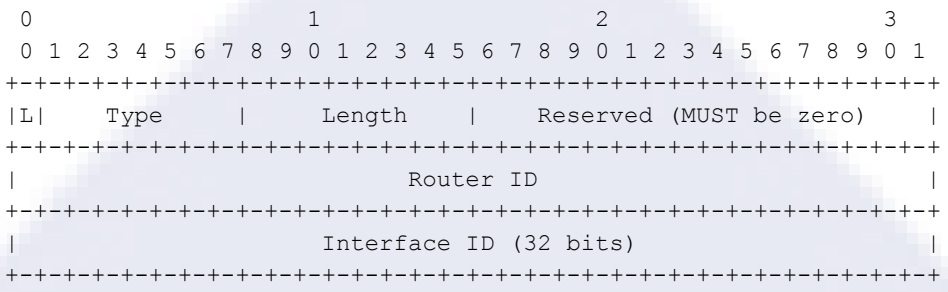
Link Identifiers is a sequence of type 3 TLVs as defined above.

6.3.2 EXPLICIT_ROUTE

The Explicit Route Object Class is 20 and C_Type 1. The EXPLICIT_ROUTE object has the following format [RFC3209]:



The sub-objects in the ERO MUST be used to select transport links that the connection will travel through. The sub-objects of Type 3 (Label) defined in [RFC3473] and Type 4 (Unnumbered interface ID) defined in [RFC3477] MUST be supported in the E-NNI. As mentioned in [OIF-ENNI2.0-SIG] Section 7.1, Table 1, E-NNI 2.0 maps the ASON SNPP ID definition into the tuple <RA ID, NodeID, IfIndex>. Note that the abstract RA ID is implied by NodeId, and does not appear explicitly in any signaling object. As a consequence, the ERO HOP Types 1, 2, and 32 are not required and not supported.



In the Type 4 (Unnumbered interface ID) sub-object, depicted above, the L bit SHALL NOT be set in any of the ERO sub-objects.

The Router ID MUST be set to "Node" ID and is a transport plane name. The Interface ID is the identifier assigned to the "link" by the transport "node". It could indicate a single link or bundled link.

Both node ID and Interface ID refer to the upstream node. This choice simplifies processing of the ERO by providing a single encoding optimized for the most common situation of the processing node being at the ingress end of a link.

An example showing the specification of Explicit Route Objects in Path messages is shown in Figure 1.

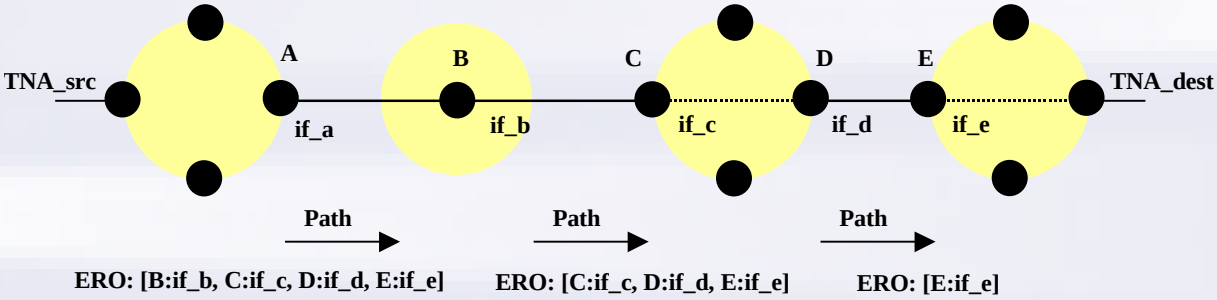


Figure 1: Example ERO Specification

Inverse Multiplexing and Transitional Link support

Figure 2 shows an example ERO specification for transitional links and inverse multiplexing. Server-layer EROs are nested in the client layer ERO. For this purpose, a new Sub-Type is defined for OIF Vendor Private EXPLICIT_ROUTE [OIF-RSVP-Ext] sub-object. For inverse multiplexing, the ERO for each member is encoded in an OIF Vendor Private EXPLICIT_ROUTE sub-object with a new sub-type. When a client layer makes an interlayer call setup request into a server layer, it removes the nested ERO from the client layer signaling and passes it as a parameter to the interlayer call setup request. For example, at node 2, the Nested_ERO_1 is removed from the Client layer ERO and passed to node 3 in the interlayer call setup request. A server layer that receives a nested ERO that consists of a list of INV_MUX_EROs will parse the list and use one ERO per connection it establishes. For example, at node 5, the Nested_ERO_2 is parsed and two INV_MUX_EROs are extracted. Each ERO is used to establish a connection.

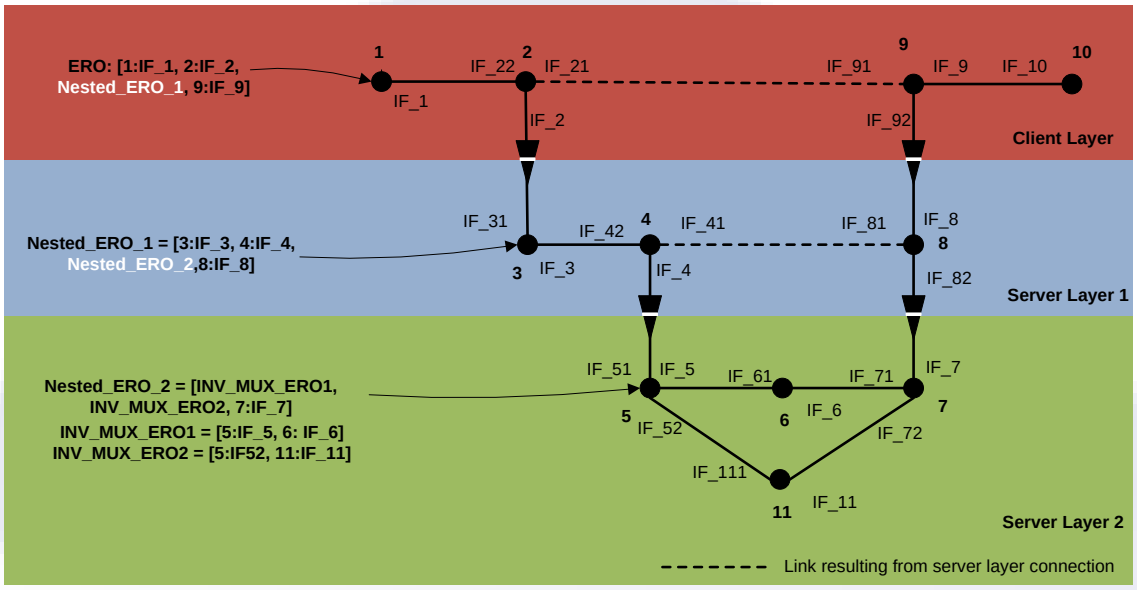


Figure 2: Inverse Multiplexing and Nesting ERO Representations

Nested and inverse multiplexing ERO sub-objects follow [OIF-RSVP-Ext] format for OIF Vendor Private EXPLICIT_ROUTE sub-objects, introducing a new sub-type as described below that contains a pointer to an OIF_VENDOR_PRIVATE_ERO sub-object in the OIF_VENDOR_PRIVATE_EXTENSION_TYPE 3 object. The OIF_VENDOR_PRIVATE_ERO_ID is an identifier that is unique in the scope of the PATH message in which it is sent and it identifies an OIF_VENDOR_PRIVATE_ERO sub-object. The OIF_VENDOR_PRIVATE_ERO sub-object is described in Section 6.3.12.2.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type (124)    |     Length    |   SMI Enterprise Code (26041) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SMI Enterpr. Code (continued) | OIF_VENDOR_PRIVATE_ERO_ID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

6.3.3 Record Route Object

The RECORD_ROUTE object is defined in [RFC3209]. The object contains a series of variable-length data items called sub-objects. Two of the sub-objects defined by the IETF for the RRO object MAY be used in E-NNI 2.0 signaling (Table 7).

Table 7: RRO Sub-objects

Sub-object	Type	Contents
Label	0x03	Use is as defined in [RFC3209] and extended in [RFC3473].

Unnumbered IPv4	0x04	Use is as defined in [RFC 3477]. NodeID is used for RouterID.

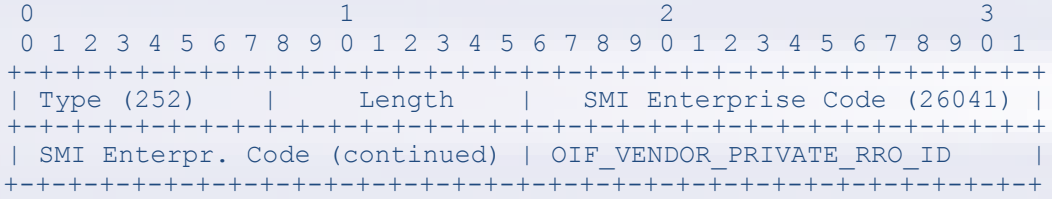
All other RRO sub-objects MUST NOT be used in E-NNI 2.0 signaling.

The RRO object syntax is designed such that, with minor changes, the whole object can be used as input to the EXPLICIT_ROUTE object. Consequently, the RRO objects MAY identify a link by specifying only the *upstream* link end, only the *downstream* link end, or by specifying *both* link ends.

Inverse Multiplexing and Transitional Link support

RRO specification for transitional links and inverse multiplexing is aligned with the ERO specification. Server layer RROs are nested in the client layer RRO. When a server layer returns an interlayer call setup indication to a client layer, it includes an RRO gathered by the server layer signaling, provided an RRO was requested by the client layer. For example, at node 3, an RRO containing similar information as contained in Nested_ERO_1 is returned to node 2 in the interlayer call setup indication. An inverse multiplexing server layer combines the INV_MUX_RROs for all of its connections into a single RRO that is returned to the client layer in the interlayer call setup indication. For example, at node 5, a nested RRO that contains a list of INV_MUX_RROs is returned to node 4 in the interlayer call setup indication.

Nested and inverse multiplexing RRO sub-objects follow [OIF-RSVP-Ext] format for OIF Vendor Private RECORD_ROUTE sub-objects, introducing a new sub-type as described below that contains a pointer to an OIF_VENDOR_PRIVATE_RRO sub-object in the OIF_VENDOR_PRIVATE_EXTENSION_TYPE 3 object. The OIF_VENDOR_PRIVATE_RRO_ID is an identifier that is unique in the scope of the PATH or RESV message in which it is sent. The OIF_VENDOR_PRIVATE_RRO sub-object is described in Section 6.3.12.3.



6.3.4 Generalized UNI Object

This object is used to specify the calling and called party identifiers and other call attributes. They are requested by the user through the control plane (UNI signaling) or by the management plane (configured at the network side). The attributes are used by the called party call controller and may be used by the destination network call controller for call validation. While the majority of sub-objects must be transmitted by E-NNI signaling without any alteration, some sub-objects may be translated when the E-NNI is used between carriers. As an example, the Service Level value may be translated when going from one domain to another.

The contents of a GENERALIZED_UNI object are a series of variable-length data items. For future compatibility, the Type and Sub-Type values are assigned according to the rules pertaining to RSVP objects ([RFC2205], Section 3.10), but from their own number space. The treatment of future Type and Sub-Type values is the same as specified for RSVP Class-Num and C-Type, respectively. If an error message is to be sent due to an unrecognized Type or SubType value, a node SHOULD use the error code "unknown Class-Number" or "unknown C-Type with known Class-Number" and the error value set to the Class-Number and C-Type of the GENERALIZED_UNI object.

Note that in E-NNI 2.0, the EGRESS_LABEL (Type 4 sub-object) or SPC_LABEL value is set to zero when a label value is not required. The particular egress label type provided does not specify whether signaling is active on the egress interface or not. For multilayer endpoints, the SPC_LABEL MUST be used when an egress logical port identifier is required.

The GENERALIZED_UNI object is carried in the Path message of a recovery connection. The source and destination TNAs (and the service level, etc.) are the end-to-end ones. They generally do not identify the recovery connection endpoints in its recovery domain. See section 6.7.2 for more details about the identification of the recovery connection endpoints.

6.3.5 RESV_CONFIRM

The IPv4 RESV_CONFIRM (class = 15, C-Type =1) object is defined in [RFC2205]. IPv4 RESV_CONFIRM MUST be supported.

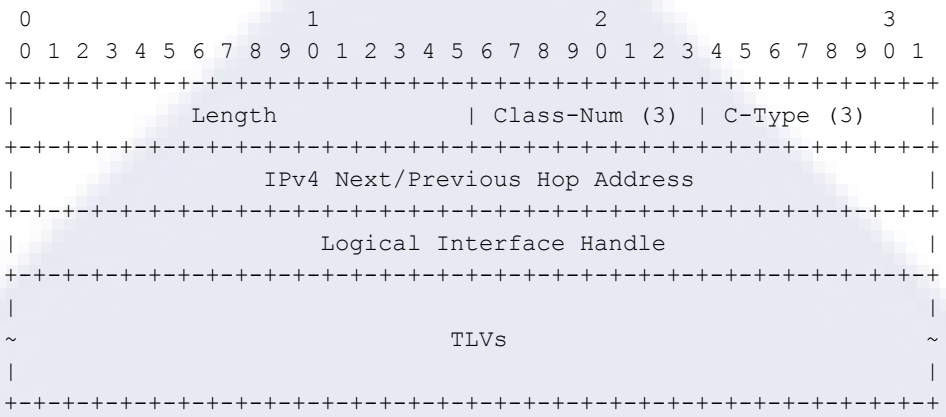
In E-NNI signaling, the receiver address MUST be set to the SC PC ID of the downstream E-NNI (Identifier of eNNI-D).

6.3.6 RSVP_HOP

There are two types of RSVP_HOPs supported at the E-NNI: IF_ID RSVP_HOP (C-Type 3) and IPv4 RSVP_HOP (C-Type 2) The IPv4 RSVP_HOP MUST be used at the VCAT layer. IF_ID RSVP_HOP MUST be used for all other layers. Use of the Type 2 TLV in transport networks is not supported and is for further study.

6.3.6.1 IF_ID RSVP_HOP

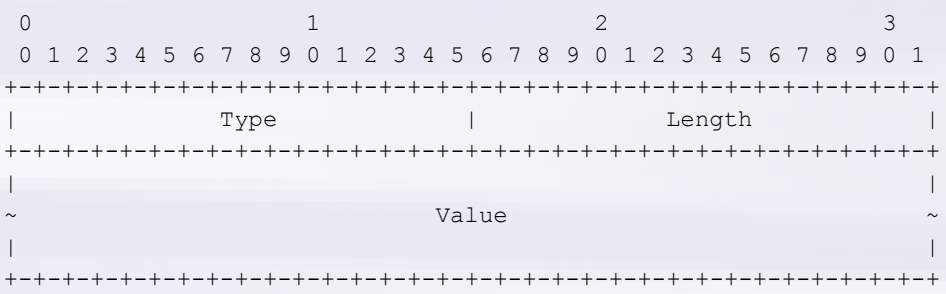
The format of this object is provided in [RFC3473]:



The IPv4 Next/Previous Hop Address field SHOULD be set to the SC PC ID (Identifier of the eNNI-U or eNNI-D) corresponding to this link. In Path messages, it contains the corresponding SC PC ID of the upstream domain and in Resv messages the corresponding SC PC ID of the downstream domain.

As described in [RFC2205], a node receiving an LIH (Logical Interface Handle) in a Path message saves its value and returns it in the HOP objects of subsequent Resv messages sent to the node that originated the LIH. Other than satisfying this requirement, *a node receiving an LIH should not expect to receive any specific value in the LIH*. Nodes MAY include a value equal to that of the interface ID (see below) or 0.

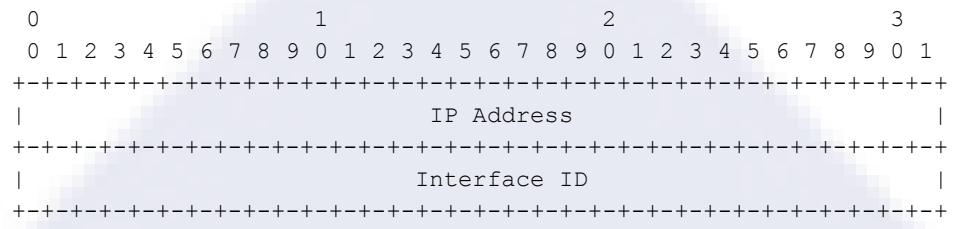
The format of TLVs is defined in [RFC3471]:



The Type 3 MUST be supported:

Type	Length	Format	Description
3	12	See below	IF_INDEX (Interface Index)

For Type 3 the Value field has the format:



The IF_ID_RSVP_HOP object MUST be used to select the transport link where a connection's resources should be allocated. The IP Address field should be set to the node ID corresponding to this link.

- For a unidirectional LSP, a downstream data link MUST be indicated.
- For bidirectional LSPs, a common downstream and upstream data link is normally indicated. In the special case where a bidirectional LSP traverses a bundled link, it is possible to specify a downstream data link that differs from the upstream data link. When two RSVP_HOP sub-objects are required, the Type 3 sub-object MUST be used as follows:
- The first sub-object MUST represent the downstream data link
- The second sub-object MUST represent the upstream data link

The interface ID carries the interface identifier for a single link or a bundled component link. The mapping of Interface IDs should be maintained at both the eNNI-U and eNNI-D i.e., the local and remote interface ID might not be identical.

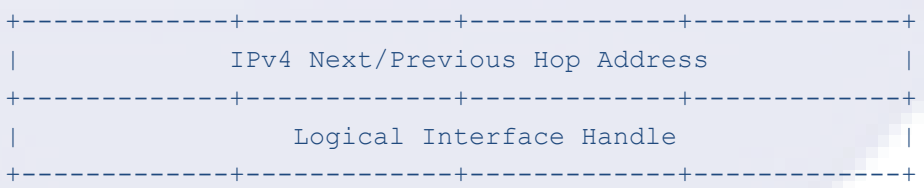
Note: Use of node identifiers beyond IP addresses may be desirable (for example specifying nodes by a name). Support for this capability is for further study.

6.3.6.2 IPv4 RSVP_HOP

For the inverse multiplexing case, the IPv4 RSVP_HOP is used and an OIF vendor private extension carries the multiple interface ids as described in section 6.3.11.1.

The regular IPv4 RSVP_HOP [RFC2205] is used as follows:

```
IPv4 RSVP_HOP object: Class = 3, C-Type = 1
```

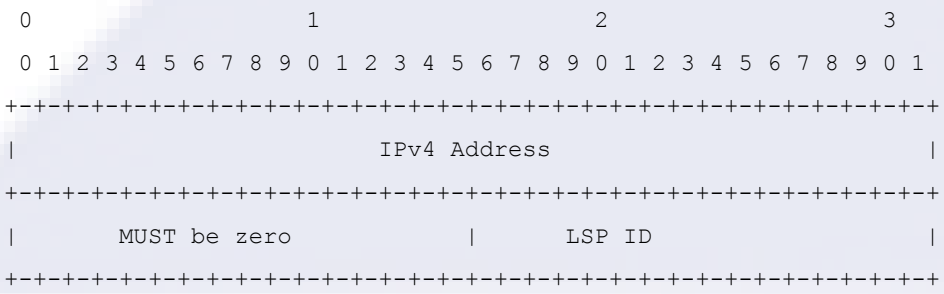


The IPv4 Next/Previous Hop Address field SHOULD be set to the SC PC ID (Identifier of the eNNI-U or eNNI-D) corresponding to interface ids that are inverse multiplexed. The LIH should be treated as for IF_ID_RSVP_HOP described above.

6.3.7 SENDER_TEMPLATE

The LSP_TUNNEL_IPv4 object (C-Type = 7) is defined in [RFC3209]. The LSP_TUNNEL_IPv4 MUST be supported, and MUST be used in SENDER_TEMPLATE and FILTER_SPEC across E-NNI interfaces. The LSP_TUNNEL_IPv4 object has the following format [RFC3209]:

- LSP_TUNNEL_IPv4 object : Class = 11, C-Type = 7



IPv4 Address: This MUST be set to the SC PC ID of the upstream E-NNI (Identifier of eNNI-U).

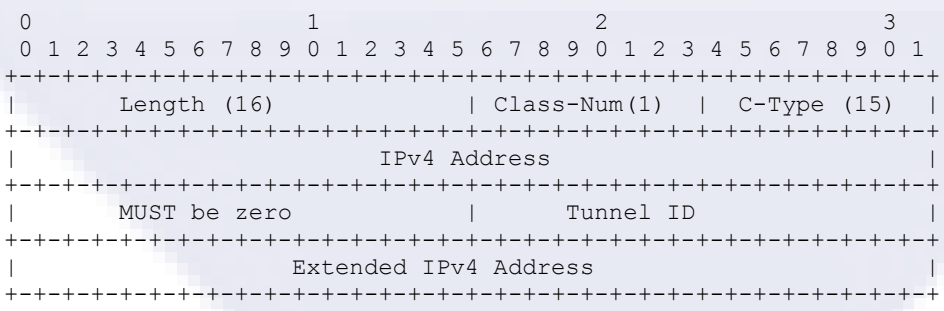
LSP ID: A 16-bit identifier used in the SENDER_TEMPLATE and the FILTER_SPEC.

The combination of the LSP_TUNNEL_IPv4_SENDER_TEMPLATE object and E-NNI_IPv4_SESSION object MUST uniquely identify a connection at a local E-NNI. In the case of connection bandwidth modification using the make-before-break procedure, the LSP_TUNNEL_IPv4_SENDER_TEMPLATE LSPID will change during the duration of the connection; all other parameters specified by these two objects remain unmodified. Otherwise, these two objects remain unmodified for the duration of the connection. An unrecognized connection ID SHOULD result in an error message with error code “Routing Problem: Invalid/Unknown Connection ID”.

6.3.8 SESSION

The E-NNI_IPv4 SESSION object (C-Type = 15) is defined in [RFC 3474]. The E-NNI_IPv4 SESSION MUST be supported across E-NNI interfaces. The SESSION object with C-Type = 15 has the following format:

- E-NNI_ IPv4_SESSION object: Class = 1, C-Type = 15



IPv4 Address: This MUST be set to the SC PC ID of the downstream E-NNI (eNNI-D).

Tunnel ID: A 16-bit identifier, assigned by the sender of the Path message. This ID remains constant during the life of a connection.

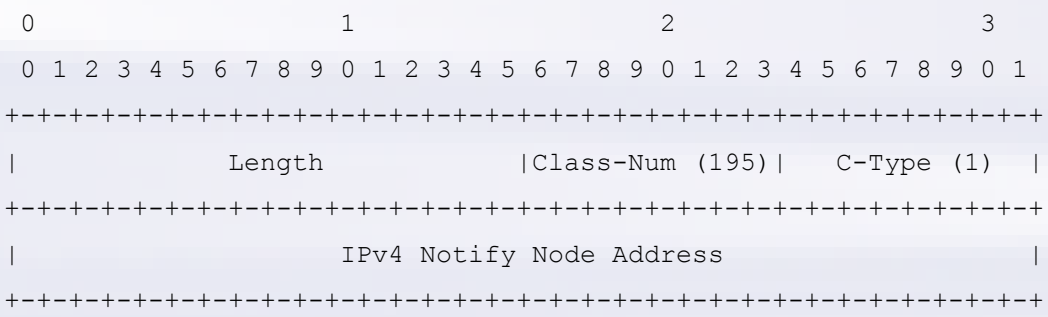
Extended IPv4 address: This MUST be set to the SC PC ID of the upstream E-NNI (Identifier of eNNI-U).

The combination of the LSP_TUNNEL_IPv4_SENDER_TEMPLATE object and E-NNI_IPv4_SESSION object MUST uniquely identify a connection at a local E-NNI. In the case of connection bandwidth modification using the make-before-break procedure, the LSP_TUNNEL_IPv4_SENDER_TEMPLATE LSPID will change during the duration of the connection; all other parameters specified by these two objects remain unmodified. Otherwise, these two objects remain unmodified for the duration of the connection.

An unrecognized connection ID SHOULD result in an error message with error code "Routing Problem: Invalid/Unknown Connection ID".

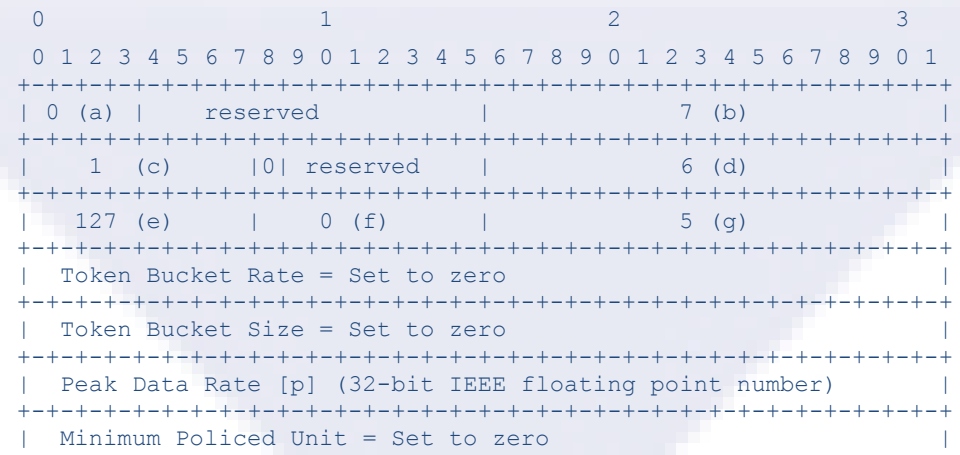
6.3.9 NOTIFY_REQUEST

The Notify Node Address field contains the SC PC ID of the E-NNI node that generates the object:



6.3.10 INTSERV_ TSPEC and FLOWSPEC

For SONET/SDH VCAT layer signaling, the PATH message MUST include the INTSERV_TSPEC [RFC2210] and the RESV message MUST include the INTSERV_FLOWSPEC. The requested VCAT data rate MUST be used to fill the Peak Data Rate for the TSPEC and FLOWSPEC objects. All other fields of the INTERV_TSPEC and INTSERV_FLOWSPEC MUST be filled as follows:



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Maximum Packet Size  = Set to zero                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

6.3.11 OIF_VENDOR_PRIVATE_EXTENSION_TYPE_1

OIF_VENDOR_PRIVATE_EXTENSION_TYPE_1 [OIF-RSVP-Ext] carries the OIF_INV_MUX_IF_ID , OIF_INV_MUX _TSPEC and OIF_INV_MUX_FLOWSPEC.

6.3.11.1 OIF_INV_MUX_IF_ID

The OIF_INV_MUX_IF_ID is the first level sub-object used to carry the list of interface identifiers that represent the components that carry the inverse multiplexed traffic. The first level sub-object includes a list of second level sub-objects defined below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             | Class-Num (1) |   C-Type (1)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                  Second Level Sub-objects                     ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Inverse Multiplexing Interface Identifier List (INV_MUX_IF_ID_LIST) second level sub-object

Each sub-object includes a list of interface identifiers for the specified signal type. If all member constituents are of the same signal type, a single second level sub-object is present. For example, a VC-4-7v would have a single INV_MUX_IF_ID_LIST second level sub-object with a Signal Type of VC-4 and 7 interface identifiers.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             | Class-Num (1) |   C-Type (1)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Signal Type |               Reserved                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Interface ID                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Interface ID                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Signal Type:

For SONET/SDH VCAT (as indicated by an Encoding Type set to SONET/SDH in the Generalized Label Request), the Signal Type represents the constituent signal type as defined in [RFC4606].

Reserved:

The bits SHOULD be set to 0 when sending, ignored upon receipt.

Interface ID:

The nth sub-object MUST represent the nth member data link added to the inverse multiplexing group. Each sub-object's interface id corresponds to a TCP returned by the server layer corresponding to a member connection.

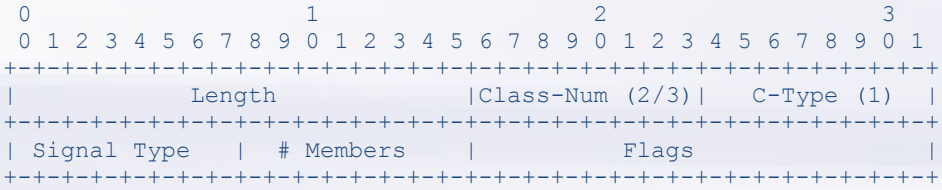
The mapping of interface IDs should be maintained at both the eNNI-U and eNNI-D i.e., the local and remote interface ID might not be identical. Interface identifiers are specified from the viewpoint of the sender of the Path message. The same identifiers are specified in the Resv message.

In order to allow VCAT group size changes, the object is allowed to be modified using non-disruptive connection modification signaling procedures described in section 6.4.2.3.

The number of Interface IDs MUST correspond to the number of member field of OIF_INV_MUX_TSPEC/FLOWSPEC.

6.3.11.2 OIF_INV_MUX_TSPEC/FLOWSPEC

The OIF_INV_MUX_TSPEC (Class-Number 2) is the first level sub-object used to carry the inverse multiplexing traffic parameters in the PATH message. The OIF_INV_MUX_FLOWSPEC (Class-Number 3) is the first level sub-object used to carry the inverse multiplexing traffic parameters in the RESV message.

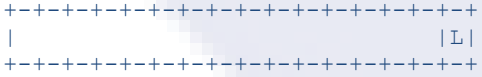


The Signal type takes one of the following values:

Value	Signal Type (Inverse Multiplexing Type)
1	VCAT

The Number of members represents the number of constituents that are inversely multiplexed and included in the INV_MUX_IF_ID_LIST.

For a signal type of VCAT (1), the following flags are defined:



Flags:

L: LCAS is required. If LCAS is not available, the request MUST be rejected.

6.3.11.3 OIF_RECOVERY_STACK object

[OIF-ENNI-REC-AM-01.0] identified the need to carry per recovery domain recovery information as summarized below:

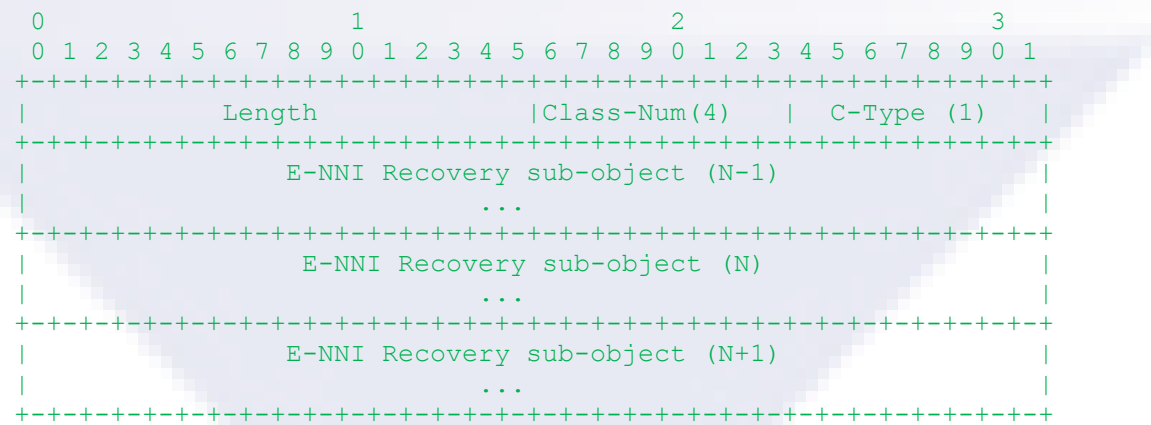
- At an E-NNI interface within a recovery domain, the PROTECTION and ASSOCIATION objects used for signaling of the recovery mechanism chosen in that recovery domain are encapsulated into an E-NNI Recovery sub-object. This applies both to the signaling of the working and the recovery connections.
- For shared-mesh restoration, a PRIMARY_PATH_ROUTE object may be included during the signaling of a recovery connection: it is also encapsulated into an E-NNI Recovery sub-object. If intermediate nodes have to make a decision about resource sharing between recovery connections, then this parameter MUST be provided. If the recovery connection Explicit Route specifies all resources down to the labels (it may have been computed by a PCE for instance), the Explicit Route itself implies whether resources are shared or not with other recovery connections, and the PRIMARY_PATH_ROUTE object is therefore not needed.

Such recovery information is carried in a first-level sub-object of the OIF OIF_VENDOR_PRIVATE_EXTENSION_TYPE_1 vendor private object, as defined in [RSVP-PVT-EXT-01.0]. This first-level sub-object is called OIF_RECOVERY_STACK object. Its class number is 4 and its C-Type 1.

The OIF_RECOVERY_STACK object may be encoded in Path and Resv messages. At most one E-NNI_RECOVERY_STACK object should be contained in such messages. Subsequent E-NNI_RECOVERY_STACK object instances should be ignored.

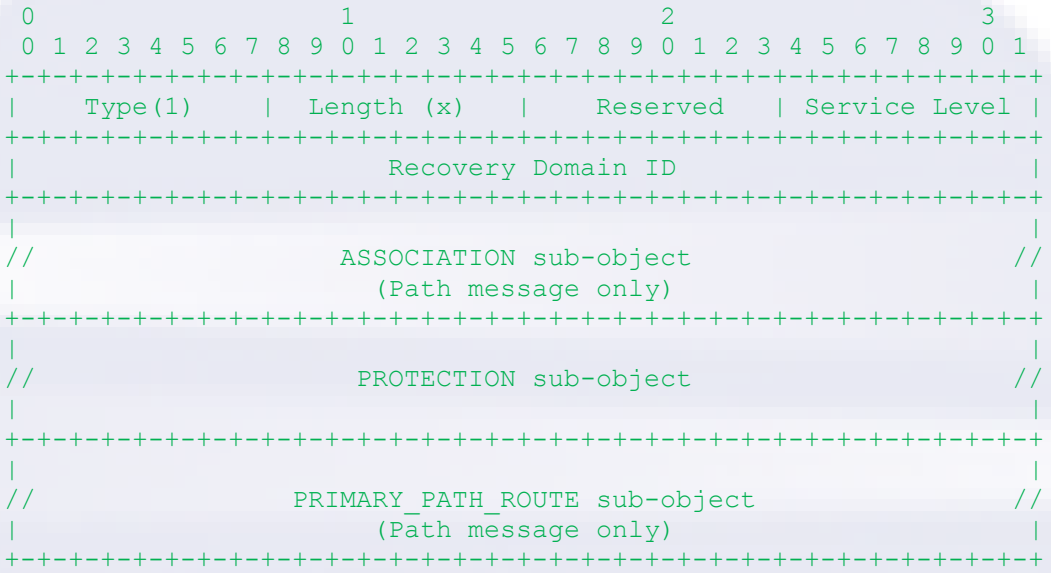
The OIF_RECOVERY_STACK object contains one or more E-NNI recovery sub-object.

As detailed in section 6.7.1, the OIF_RECOVERY_STACK object is handled as a stack of E-NNI Recovery sub-object. One E-NNI Recovery sub-object is pushed on the stack every time the connection enters a nested recovery domain. The E-NNI Recovery sub-object is popped from the stack when the connection exits that recovery domain. When push on the stack, an E-NNI Recovery sub-object is added at the beginning of the OIF_RECOVERY_STACK object, as depicted below:

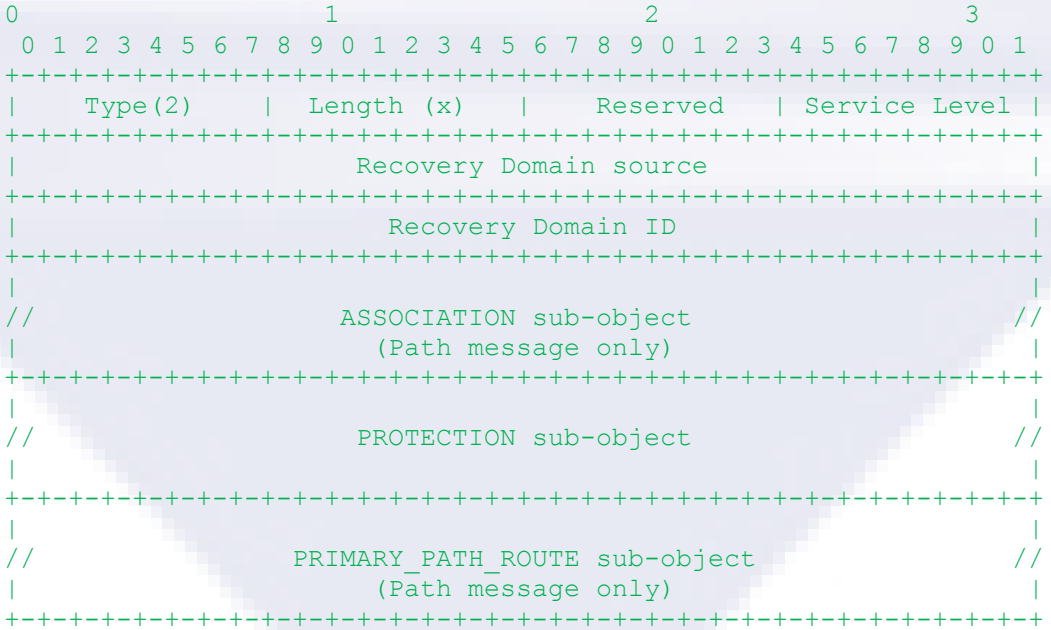


The OIF_RECOVERY_STACK object MUST carry the same stack of E-NNI Recovery sub-objects in Path and Resv messages. Note that the E-NNI Recovery sub-object associated to a given recovery domain in the stack may be different in Path and Resv messages (some sub-sub-objects, e.g. in this Amendment, ASSOCIATION are not supported in the RECOVERY_STACK of the Resv messages).

The E-NNI Recovery sub-object (TLV) for recovery domains scoped to a Routing Area has been assigned type 1. Its format is depicted below:



The E-NNI Recovery sub-object (TLV) for recovery domains scoped to an E-NNI interface has been assigned type 2. Its format is depicted below:



The fields are filled with the following values:

Recovery Domain Source (type 2 E-NNI Recovery sub-object only)

This field is set to the eNNI-U signaling controller ID.

Recovery Domain ID: 32 bits

The Recovery Domain ID field identifies the recovery domain to which the E-NNI Recovery sub-object is referring to (see E-NNI Recovery amendment).

In a type 1 E-NNI Recovery sub-object, it is set to a routing area ID.

In a type 2 E-NNI Recovery sub-object, it is set to a 32 bit integer assigned by the eNNI-U (identified in the Recovery Domain Source field).

(Domain Local) Service Level: 8 bits

This field specifies the desirable service expected from lower nested recovery domains. Its purpose is to help coordinating the recovery mechanisms chosen in multiple nested recovery domains.

It should be noted that the DIN of a nested domain (N), when receiving an OIF_RECOVERY_STACK object from its upstream peer in domain (N+1), should only consider the Service Level field of the first E-NNI Recovery sub-object (the latest added to the stack), in order to choose (policy-driven) a recovery mechanism in domain (N).

ASSOCIATION object:

See section 6.3.11.3.2.

Encoding of the ASSOCIATION object within an E-NNI Recovery sub-object is mandatory in a Path message when an end-to-end recovery mechanism has been selected within a recovery domain boundary.

When no end-to-end recovery mechanism has been selected within a recovery domain boundary, but the DIN wants to enforce using links with a specific link-protection type, an E-NNI Recovery sub-object is required for that recovery domain (to specify that link-protection type in the PROTECTION object). However, in such a case, no ASSOCIATION object may be encoded.

It should not be encoded in a Resv message

PROTECTION object:

See section 6.3.11.3.1.

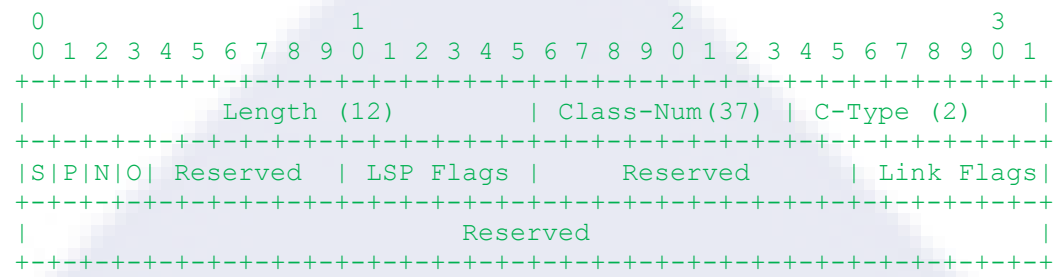
PRIMARY_PATH_ROUTE object:

See section 6.3.11.3.3

6.3.11.3.1 PROTECTION object

This amendment does not propose any change to the PROTECTION object defined in [RFC4872]. Its format is used without change. It is provided here for reference.

The format of the PROTECTION Object (Class-Num = 37, C-Type = 2) is as follows:



Secondary (S): 1 bit

When set to 1, this bit indicates that the requested LSP is a secondary LSP. When set to 0 (default), it indicates that the requested LSP is a primary LSP.

Protecting (P): 1 bit

When set to 1, this bit indicates that the requested LSP is a protecting LSP. When set to 0 (default), it indicates that the requested LSP is a working LSP. The combination, S set to 1 with P set to 0 is not valid.

Notification (N): 1 bit

When set to 1, this bit indicates that the control plane message exchange is only used for notification during protection switching. When set to 0 (default), it indicates that the control plane message exchanges are used for protection-switching purposes. The N bit is only applicable when the LSP Protection Type Flag is set to either 0x04 (1:N Protection with Extra-Traffic), or 0x08 (1+1 Unidirectional Protection), or 0x10 (1+1 Bidirectional Protection). The N bit MUST be set to 0 in any other case.

Operational (O): 1 bit

When set to 1, this bit indicates that the protecting LSP is carrying the normal traffic after protection switching. The O bit is only applicable when the P bit is set to 1, and the LSP Protection Type Flag is set to either 0x04 (1:N Protection with Extra-Traffic), or 0x08 (1+1 Unidirectional Protection) or 0x10 (1+1 Bidirectional Protection). The O bit MUST be set to 0 in any other case.

Reserved: 5 bits

This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt. These bits SHOULD be passed through unmodified by transit nodes.

LSP (Protection Type) Flags: 6 bits

Indicates the desired end-to-end LSP recovery type. A value of 0 implies that the LSP is "Unprotected". The following values are defined. All other values are reserved.

0x00 Unprotected

0x01 (Full) Rerouting

0x02 Rerouting without Extra-Traffic

0x04 1:N Protection with Extra-Traffic (not used in this implementation agreement)

0x08 1+1 Unidirectional Protection

0x10 1+1 Bidirectional Protection

Only one value SHOULD be set at a time. However, to support the combination of soft or hard-rerouting with 1+1 unidirectional or bidirectional protection, and with shared-mesh restoration, this implementation agreement allows the following values for the LSP Protection Type Flags:

0x09 Soft or hard-rerouting combination with 1+1 unidirectional protection

0x11 Soft or hard-rerouting combination with 1+1 bidirectional protection

0x03 Soft or hard-rerouting combination with shared-mesh restoration.

Reserved: 10 bits

This field is reserved. It MUST be set to zero on transmission and MUST be ignored on receipt. These bits SHOULD be passed through unmodified by transit nodes.

Link Flags: 6 bits

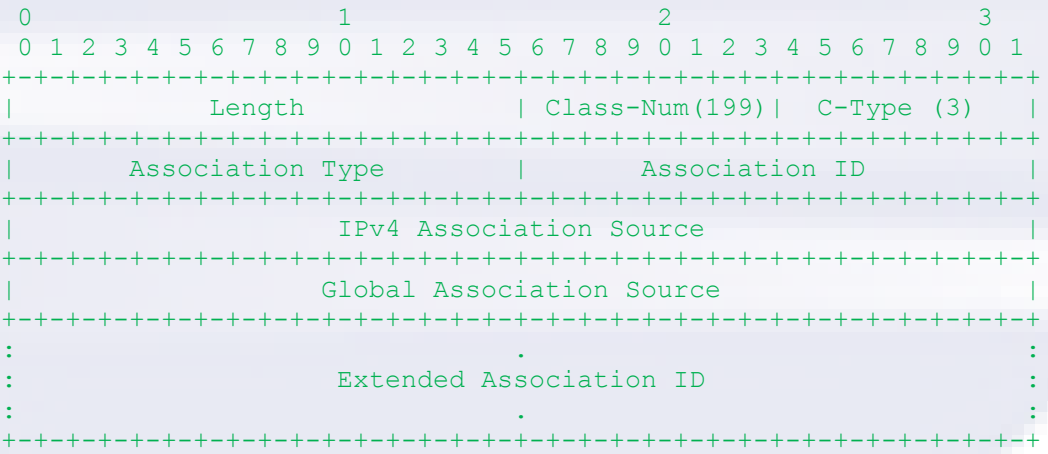
Indicates the desired link protection type (see [RFC3471]).

Reserved field: 32 bits

Encoding of this field is detailed in [RFC4873]. It SHOULD be set to zero on transmission and SHOULD be ignored on receipt. These bits SHOULD be passed through unmodified by transit nodes.

6.3.11.3.2 ASSOCIATION object

This amendment uses the Extended IPv4 ASSOCIATION object format defined in [RFC6780].



The fields are filled with the following values:

Association Type: 16 bits

Indicates the type of association being identified. Note that this value is considered when determining association.

The Association Type MUST be set to 1 (Recovery; see section 6.7.2) or to 2 (Resource Sharing; see section 6.7.3).

Association ID: 16 bits

It carries an association identifier, which is assigned by the Association Source.

Association Source: 32 bits

Is set to the DIN controller SC PC ID.

Note that although this amendment specifies using the Extended IPv4 ASSOCIATION object format (i.e., 32 bits Association Source), it does not make any assumption about the SCN: it may be an IPv6 SCN.

Global Association Source: 32 bits

This field MUST be set to 0 for intra-carrier recovery scenarios (its usage for inter-carrier recovery scenarios is for further study).

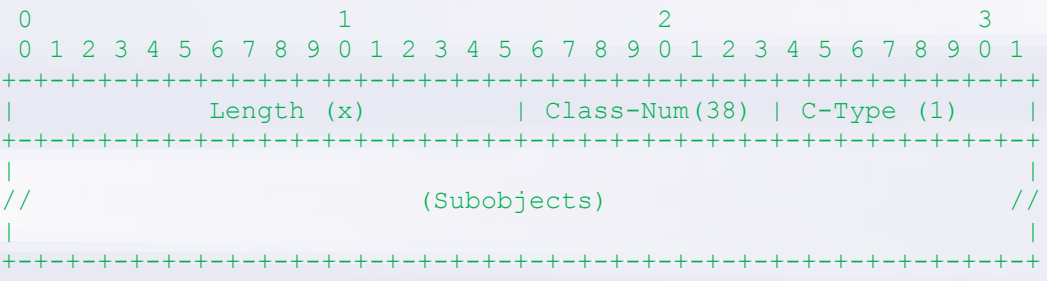
Extended Association ID:

The extended association ID field MUST have a length of zero (it MUST be omitted).

6.3.11.3.3 PRIMARY_PATH_ROUTE object

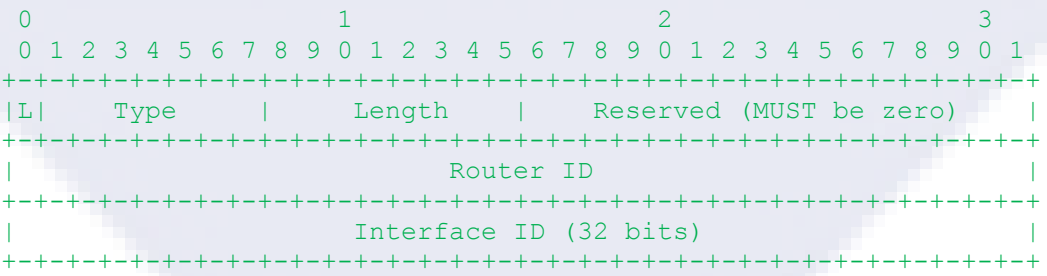
This amendment does not propose any change to the PRIMARY_PATH_ROUTE object defined in [RFC4872]. Its format is used without change. It is provided here for reference.

The format of the PRIMARY_PATH_ROUTE Object (Class-Num = 38, C-Type = 1) is as follows:



The contents of a PRIMARY_PATH_ROUTE object are a series of variable-length data items called subobjects.

The sub-objects of Type 3 (Label) defined in [RFC3473] and Type 4 (Unnumbered interface ID) defined in [RFC3477] MUST be supported in the E-NNI.



In the Type 4 (Unnumbered interface ID) sub-object, depicted above, the L bit, Router ID and Interface ID are set as specified in section 6.3.2.

Type 3 sub-objects (Label) are not required in a PRIMARY_PATH_ROUTE object. They MAY be included e.g. when a PRIMARY_PATH_ROUTE object is built by duplicating a RECORD_ROUTE object.

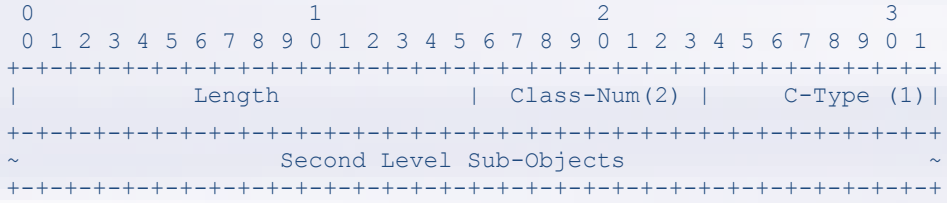
6.3.12 OIF_VENDOR_PRIVATE_EXTENSION_TYPE_3

OIF_VENDOR_PRIVATE_EXTENSION_TYPE_3 [OIF-RSVP-Ext] carries OIF_LSP_TUNNEL_INTERFACE_ID for signaling based discovery, OIF_VENDOR_PRIVATE_ERO for multilayer ERO, OIF_VENDOR_PRIVATE_RRO for multilayer RRO and OIF_ML_ADAPTATION for signaling adaptation information in the server layer.

6.3.12.1 OIF_LSP_TUNNEL_INTERFACE_ID

The OIF_LSP_TUNNEL_INTERFACE_ID should be used for signaling based discovery unless a distinct discovery protocol is used between the client layer endpoints. The OIF_LSP_TUNNEL_INTERFACE_ID MAY occur multiple times as a first level sub-TLV of the OIF_VENDOR_PRIVATE_EXTENSION_TYPE_3. Each occurrence represents a different set of client layers.

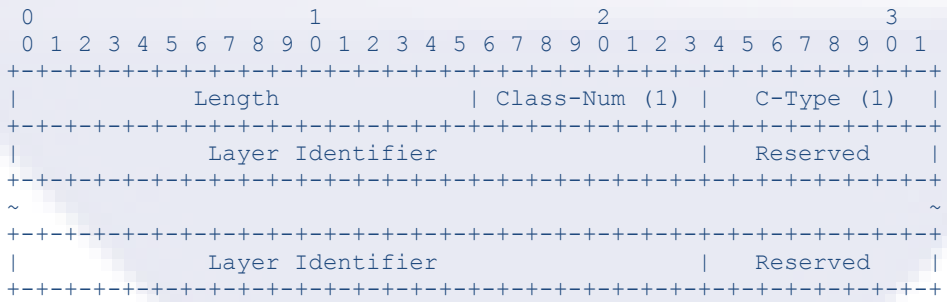
The OIF_LSP_TUNNEL_INTERFACE_ID format is indicated below. The length is variable.



The following second level sub-objects SHOULD be used for signaling based discovery unless a distinct discovery protocol is used between the client layer endpoints.

Multi-Client second level sub-object

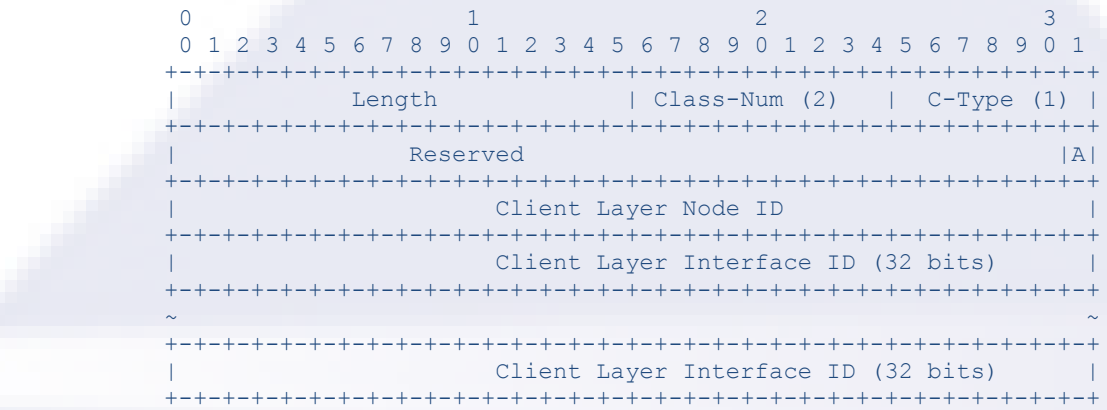
This second-level sub-object represents the set of client layers to which the first-level sub-object applies.



Layer Identifier is defined in section 6.3.15.

CLIENT_NODE_ID_IF_ID second level sub-object

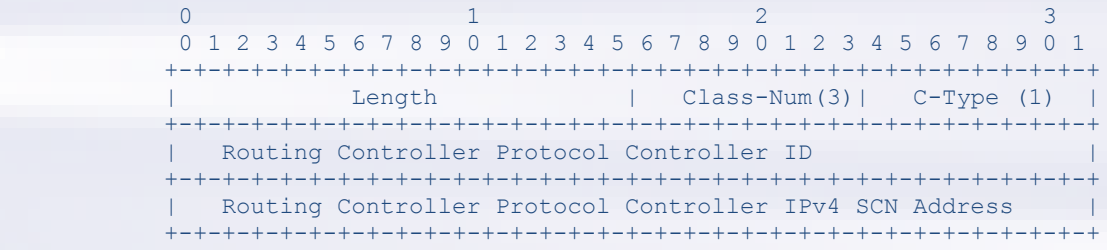
This second-level sub-object represents the node id and interface id(s) for the client layer(s) to which the first-level sub-object applies. A list of interface id(s) is included for the scenario where the VCAT constituents are established in the server layer using co-signaling approach that uses the multiplier (MT) (in SONET/SDH) value to establish multiple constituents. In this case, each constituent results in a separate client layer interface identifier in the list below.



A bit: When set to 1, indicates that the link is to be advertised in client layer routing.

ASON Routing ID second level sub-object

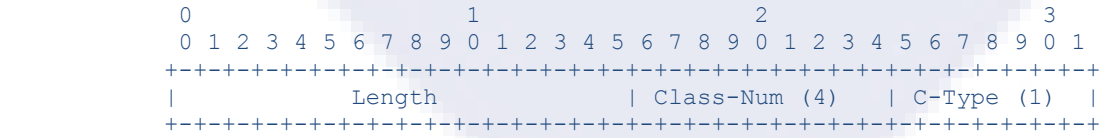
This second-level sub-object represents the set of routing identifiers used by the client layers to which the first-level sub-object applies. C-Type 1 is for OSPF-TE based routing protocol with 32-bit RC PC ID and IPv4 RC PC SCN address.



- RC PC ID – 32 bit RC PC ID of the client layers to which the first-level sub-TLV applies
- RC PC SCN Address – IPv4 SCN address of the client layers to which the first sub-TLV applies

ASON Signaling ID second level sub-object

This second-level sub-TLV represents the set of signaling identifiers used by the client layers to which the first-level sub-TLV applies. C-Type 1 is for RSVP-TE based signaling protocol with 32-bit SC PC ID and IPv4 SC PC SCN address.



```
|   Signaling Controller Protocol Controller ID                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Signaling Controller Protocol Controller IPv4 SCN Address   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- SC PC ID – 32 bit SC PC ID of the client layers to which the first-level sub-TLV applies
- SC PC SCN Address – IPv4 SCN address of the client layers to which the first sub-TLV applies

6.3.12.2 OIF_VENDOR_PRIVATE_ERO

OIF_VENDOR_PRIVATE_ERO carries the ERO for transitional links.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |Class-Num (3)  | C-Type (1)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|OIF_VENDOR_PRIVATE_ERO_ID      |    Reserved   |  Type         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Layer Identifier                    |   Adaptation  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                   Subobjects                                  ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- OIF_VENDOR_PRIVATE_ERO_ID is filled with the same value specified in the OIF Vendor Private EXPLICIT_ROUTE sub-object.
- Type:
  - 1: Nested
  - 2: Inverse Multiplexing
- Layer Identifier represents the server layer. The format is defined in section 6.3.15.
  - VCAT, Ethernet EPL and EVPL: MUST be set to 0 and ignored upon receipt
- Adaptation is defined in section 6.3.16.
- Sub-objects consist of ERO sub-objects. The ERO subobjects are allowed to be of type OIF Vendor Private EXPLICIT_ROUTE sub-object. This allows the mechanism to recurse.

6.3.12.3 OIF_VENDOR_PRIVATE_RRO

OIF_VENDOR_PRIVATE_RRO carries the_RRO for transitional links.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |Class-Num (4)  | C-Type (1)    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|OIF_VENDOR_PRIVATE_RRO_ID      |    Reserved   |  Type         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Layer Identifier               |   Adaptation  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
~                       Subobjects                              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- OIF_VENDOR_PRIVATE_RRO_ID is filled with the same value specified in the OIF Vendor Private RECORD_ROUTE sub-object.
- Type:
  - 1: Nested
  - 2: Inverse Multiplexing
- Layer Identifier represents the server layer. The format is defined in section 6.3.15.
- Adaptation is defined in section 6.3.16.
- Sub-objects consist of RRO sub-objects. The RRO subobjects are allowed to be of type OIF Vendor Private RECORD_ROUTE. This allows the mechanism to recurse.

6.3.12.4 OIF_ML_ADAPTATION

The OIF_ML_ADAPTATION should be used for signaling which adaptation and client layer to use at the tail end of the server layer connection.

The OIF_ML_ADAPTATION format is indicated below.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |  Class-Num(5) |     C-Type (1)|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Layer Identifier                |   Adaptation  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- Layer Identifier represents the client layer. The format is defined in section 6.3.15.
- Adaptation is defined in section 6.3.16.

6.3.13 VCAT Labels

The Generalized Label Request for SONET/SDH VCAT has an Encoding Type value of 0x85 and a Switching Type of TDM (100) [RFC3471].

For all VCAT layer related label objects, the label format is as defined below:

A 32-bit label filled with 0xFFFFFFFF.

6.3.14 CALL ID

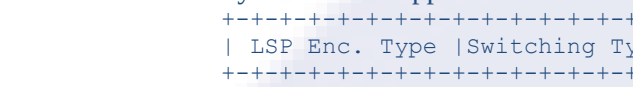
The CALL_ID format is defined in [RFC3474] with further clarifications provided in [OIF-UNI-02.0-R2-RSVP]. To guarantee uniqueness of the CALL_ID across multiple domains and multiple layers, the Source LSR address type SHOULD be set to 0x7f (vendor-specific length) [RFC3474], the length SHOULD be 8 bytes and the LSR address SHOULD be filled as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| LSP Enc. Type |Switching Type |   Signal Type |  Reserved     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SC PC ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

For backwards compatibility in single layer applications, the source LSR address MAY be filled with the SC PC ID, using the appropriate LSR address type as described in [RFC3474].

6.3.15 Layer Identifier

A Layer Identifier appears in several sub-objects and is formatted as follows:



LSP Encoding Type is set as follows:
- For SONET/SDH, excluding VCAT: 5[RFC3471]
- For SONET/SDH VCAT: 0x85
- OTN ODUk: 12 (ODUk) [RFC4328]
- Ethernet EPL and EVPL: 2 (Ethernet) [RFC3471]

Switching Type is set as follows:
- SONET/SDH, OTN ODUk, including VCAT: 100 (TDM) [RFC3471]
- Ethernet EPL: 125 (DCSC) [RFC6002]
- Ethernet EVPL: 30 (EVPL) [RFC6004]

Signal Type:
- SONET/SDH: see [RFC3946]
- OTN ODUk: see [RFC4328]
- VCAT, Ethernet EPL and EVPL: MUST be set to 0 and ignored upon receipt

6.3.16 Adaptation

Adaptation appears in several sub-objects and contains one of the following values, scoped by the server layer:

- SONET/SDH (excluding VCAT) server layer
  - GFP-F 0x01
  - GFP-T 0x02
  - VCAT (no LCAS) 0x03
  - VCAT (LCAS) 0x04
- SONET/SDH VCAT server layer
  - GFP-F 0x01
  - GFP-T 0x02
- OTN ODUk (where k = 1,2,3) server layer
  - GFP-F 0x01
  - GFP-T 0x02
  - AMP 0x05
  - BMP 0x06

6.4 RSVP-TE Signal Flows

RSVP-TE for E-NNI follows the signal flows specified in [OIF-ENNI2.0-SIG] Section 12. This section describes the RSVP-TE signal flows for connection setup, connection modification, and connection deletion.

6.4.1 Connection Setup

Figure 3 shows the setup of a connection across the E-NNI interface. Upon receiving a connection request from the network, the eNNI-U sends a Path message to the eNNI-D. The eNNI-D continues the setup request downstream. When the eNNI-D receives the setup response

indication from the network, it generates a Resv message to the eNNI-U. Within the setup indication, the egress node can request an optional confirmation message. If the eNNI-U receives the confirmation from the network, it sends a ResvConf message to the eNNI-D. The eNNI-D continues to forward the confirmation to the egress.

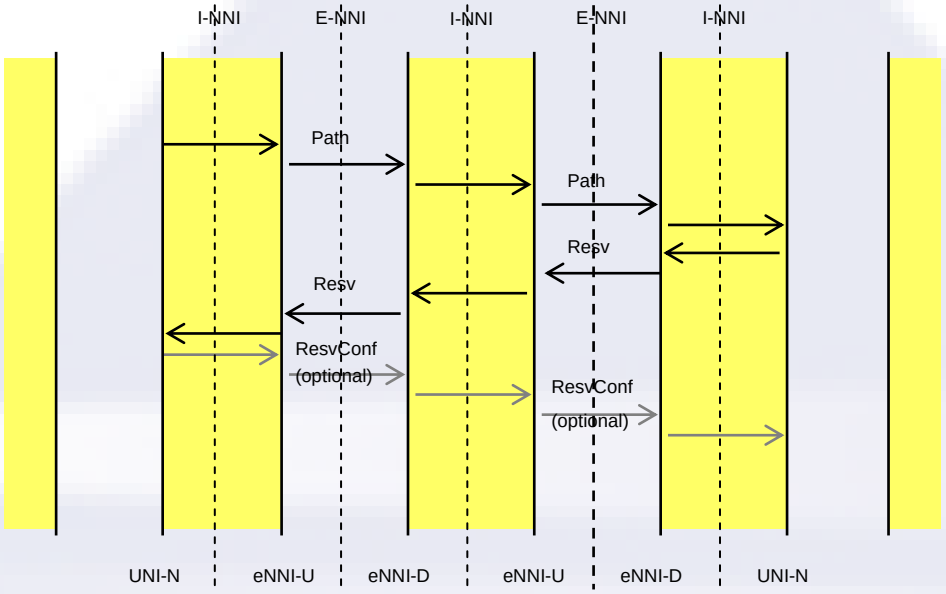


Figure 3: Basic Connection Setup Across the E-NNI

A connection setup can fail for a number of reasons including policy failure, inability to allocate resources, or destination not reachable. In the case that the eNNI-D fails the connection setup, or if the eNNI-D receives a connection setup failure indication from the network, it MUST delete its own path state and send a PathErr with the Path_State_Removed to the eNNI-U. The eNNI-U then forwards the connection failure indication towards the ingress node. This signal flow is shown in Figure 4.

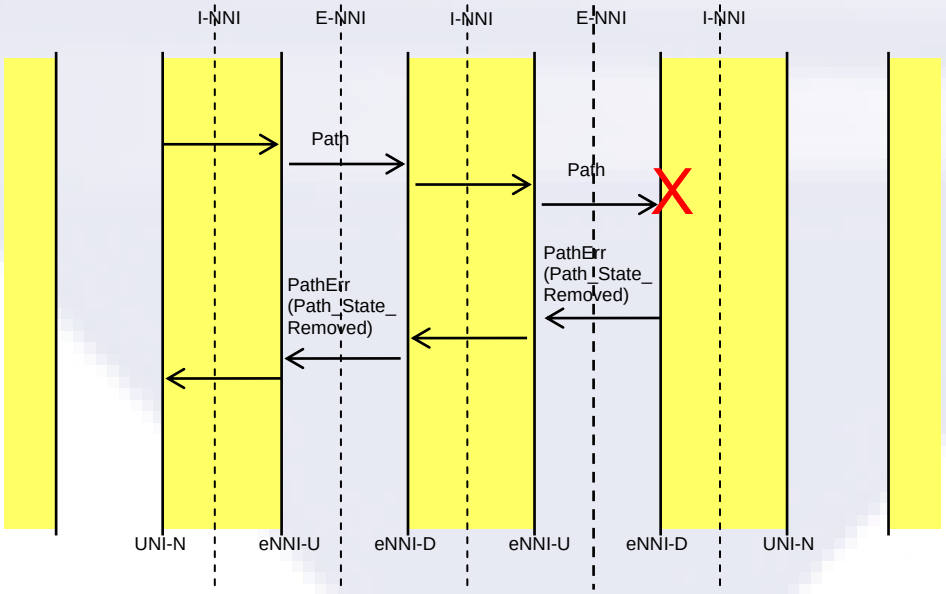


Figure 4: Connection Setup Failure

A connection setup can also fail during the indication (Resv) phase as shown in Figure 5. For instance, the label allocation can fail perhaps due to contention with another connection setup. In this case, the eNNI deletes its path state and generates PathErr with Path_State_Removed in the upstream direction and PathTear in the downstream direction.

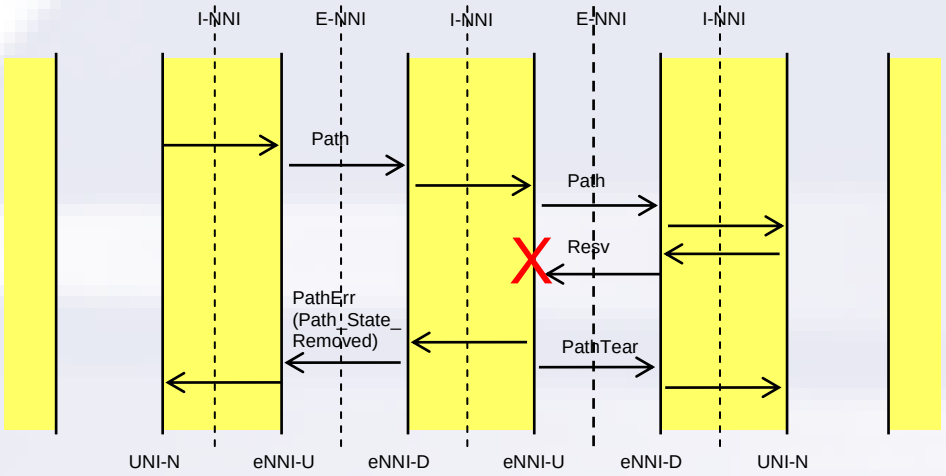


Figure 5: Connection Setup Failure during Indication

If the Path_State_Removed flag is not set in the PathErr message, then the source UNI-C or UNI-N deletes the connection explicitly. This is shown in Figure 6. When the explicit teardown request reaches the eNNI, the eNNI-U sends PathTear to the eNNI-D. A node receiving a PathTear that does not match any path state MUST acknowledge the message if the PathTear carries a MESSAGE_ID with the Ack_Desired flag set and then discard the PathTear message.

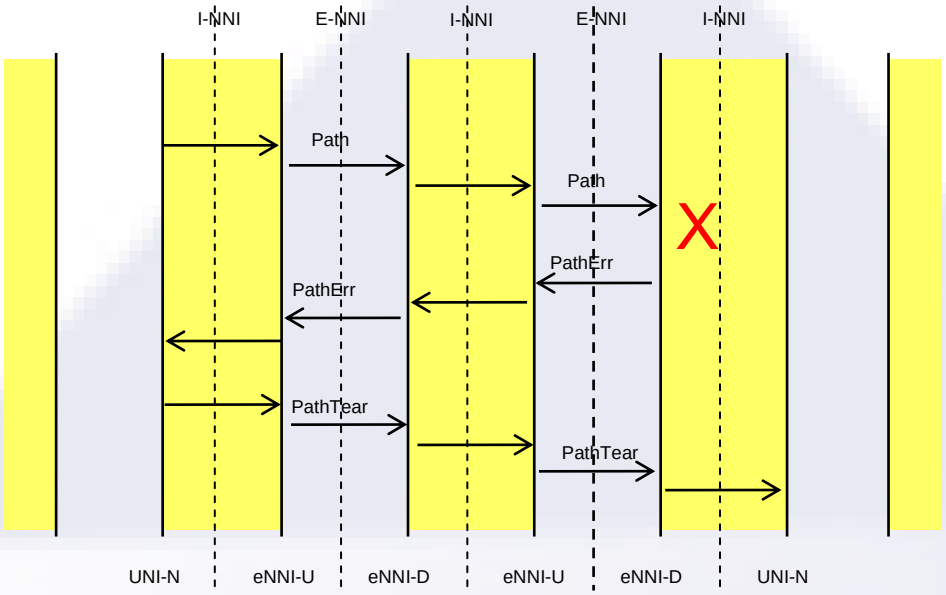


Figure 6: Connection Setup Failure without Setting the Path_State_Removed flag

6.4.2 Call Modification

A call can be modified in three ways. First, a call can be modified by adding or removing a connection to an existing call. In this case, the connection modification procedures are not used. Instead, a new connection setup request or connection release request is issued. Connections being added may follow the same route as existing connections or may be diversely routed (follow different routes). Diverse routing of connections can only be accomplished by adding connections to an existing call. The second method is to modify an existing connection within a call. The third method consists of modifications in the client/server call relationship. For example, a client layer call may be modified by increasing the number of server layer calls associated with the client layer call in the 1:n client/server relationship, e.g. a VCAT call is modified from VC-4-5v to VC-4-6v by adding a VC-4 call to the server layer and modifying the adaptation. Another example is the m:1 relationship where the number of client layer calls using resources of the same server layer call may be changed, e.g. a 100Mbps Ethernet call is using resources from a STM-16 call, then another 100Mbps Ethernet call is added and sharing the resources from the same STM-16 call.

6.4.2.1 Call Modification by Adding and Removing Connections

Adding and removing connections to an existing call can be used to modify the bandwidth of a call. A failure to add or remove a connection does not impact other connections in the call. That is, the connections remain independent of each other within the call.

Figure 7 shows a successful addition of a connection to an existing call. The call is established when the first connection is established. If the source subsequently wants to modify the call by adding another connection, it will generate a new Path message with the same CALL_ID as in the existing connection. The presence of the CALL_ID in a Path message for a new connection is used to infer that a connection is being added to the specified call. CALL_ID is used to correlate the various connections at the E-NNI nodes. Upon receiving the call modification request for this scenario, the eNNI-U will generate a new Path message with the same CALL_ID. This new Path message will have a different connection identifier (TUNNEL_ID) and a new MESSAGE_ID. If

the E-NNI node does not have an existing call state for the received CALL_ID, then the E-NNI node handles this as a new call setup request instead of a call modification.

Failure to add a connection to an existing call does not impact other connections because each connection has its own RSVP state.

The connection deletion message sequence is described in Section 6.4.3. Individual connections within a call can be deleted from the source, the destination, or from the network including from the E-NNI nodes. Each connection deletion is performed independently. A call without connections is not supported. Removal of the last connection results in the removal of the call state.

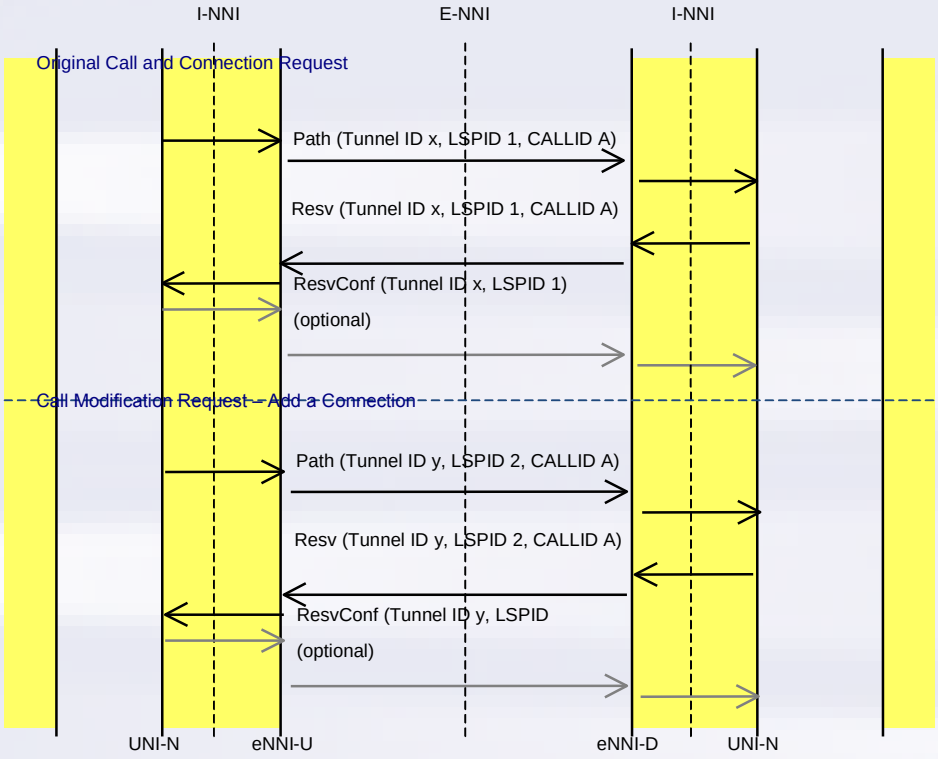


Figure 7: Successful Call Modification – Adding a Connection

6.4.2.2 Call Modification by Modification of an Existing Connection

In the second method, call modification can be supported by modifying an existing connection. E-NNI Signaling 2.0 supports the non-disruptive connection modification of the following service parameters:

- Bandwidth fields of the ETHERNET_TSPEC/FLOWSPEC (CIR, CBS, EIR, or EBS) and/or the CE-VLAN ID mapping information
- Multiplier field of the SONET_SDH_TSPEC/FLOWSPEC or G709_TSPEC/FLOWSPEC
- Number of members in VCAT call (OIF_INV_MUX_TPSEC/FLOWSPEC)

- Bandwidth for VCAT (peak data rate of the INTSERV_TSPEC/FLOWSPEC)

To support connection modification, the original connection must have been established with the Shared Explicit (SE) reservation style. This allows two or more RSVP Path states to share the same resources (such as bandwidth or CE-VLAN ID information). An E-NNI 2.0 implementation that supports non-disruptive service parameter modification SHOULD request the SE reservation style in the SESSION_ATTRIBUTE of the original Resv message. If the SE reservation style was not used in the Resv message, an E-NNI 2.0 node MUST NOT forward a non-disruptive connection modification. In this case, the E-NNI 2.0 node generates an error indication with the error code set to "Traffic Control Error: Service Unsupported".

A non-disruptive connection modification of a service parameter is illustrated in Figure 8. This same message flow is used to support modification of the bandwidth of an existing connection or modification of the CE-VLAN ID mapping to an EVC.

The call is established when the first connection is established. The eNNI-U sends a Path message to the eNNI-D to request call and connection creation. The Path request contains a SESSION_ATTRIBUTE object with the SE Style request flag set. The CALL_ID is set to the value received in the connection setup request received from the network. If a CALL_ID was not present in the connection setup request, then the eNNI-U assigns a unique CALL_ID object and adds it to the Path message (see Section 6.3.14). The eNNI-U also assigns a locally unique SESSION object for this connection and a unique LSP ID within the SENDER_TEMPLATE scoped to the SESSION object.

If the eNNI-U receives a connection modification request for this connection, it must correlate the request to an existing connection. If the connection modification request contains a CALL_ID for which there is no call state, then the eNNI-U sends an error indication "invalid/unknown call ID". Also, if there is no connection state, the eNNI-U sends an error indication "invalid/unknown connection ID". Once correlated, the eNNI-U continues the connection modification over the E-NNI. The eNNI-U generates a new Path message using the same SESSION object as the original connection but with a different LSP_ID. This allows the original and new connections to share connection resources.

On receiving the connection modification indication, the eNNI-D generates a new Resv message in the reverse direction. This is achieved by generating a new Resv message for the new Path state. It contains the FILTER_SPEC and labels of the corresponding Path message only. Continuing to refresh the previous Resv message, corresponding to the state that is awaiting teardown, until the PathErr with PATH_STATE_REMOVED flag set has been received is RECOMMENDED.

The eNNI-U MUST send a new ResvConf message if it receives a modification confirmation from the network.

At this point, there should be a removal request from the network for the original connection. Upon receipt of the removal request from the network, the eNNI-U should generate a Path message for the original connection with the Delete and Reflect (D&R) bits set in the ADMIN_STATUS object. This results in the graceful removal of the RSVP Path state at the eNNI-U and eNNI-D when the eNNI-D responds with a PathErr message with the PATH_STATE_REMOVED flag set.

The deletion of the original Path state causes the removal of the corresponding Resv state after the PathErr message with the PATH_STATE_REMOVED flag set has been received. Then the Resv corresponding to the deleted state will stop being refreshed.

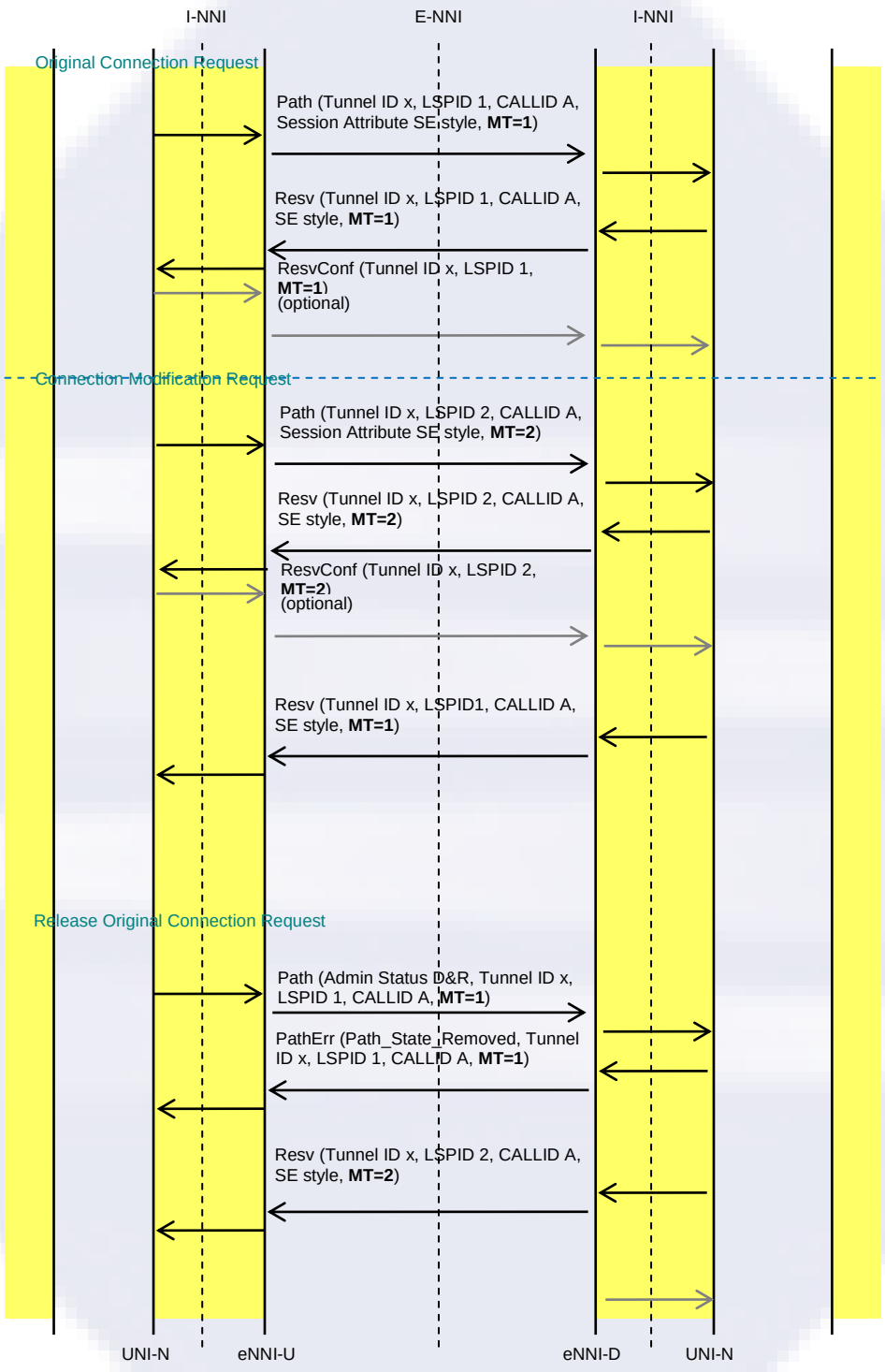


Figure 8: Successful Connection Modification

A failure to increase a connection bandwidth SHOULD result in a PathErr being sent for the Path message requesting more bandwidth. This MUST NOT impact the existing connections, other Path messages, or RSVP states. Figure 9 illustrates a failure to increase the bandwidth of a connection.

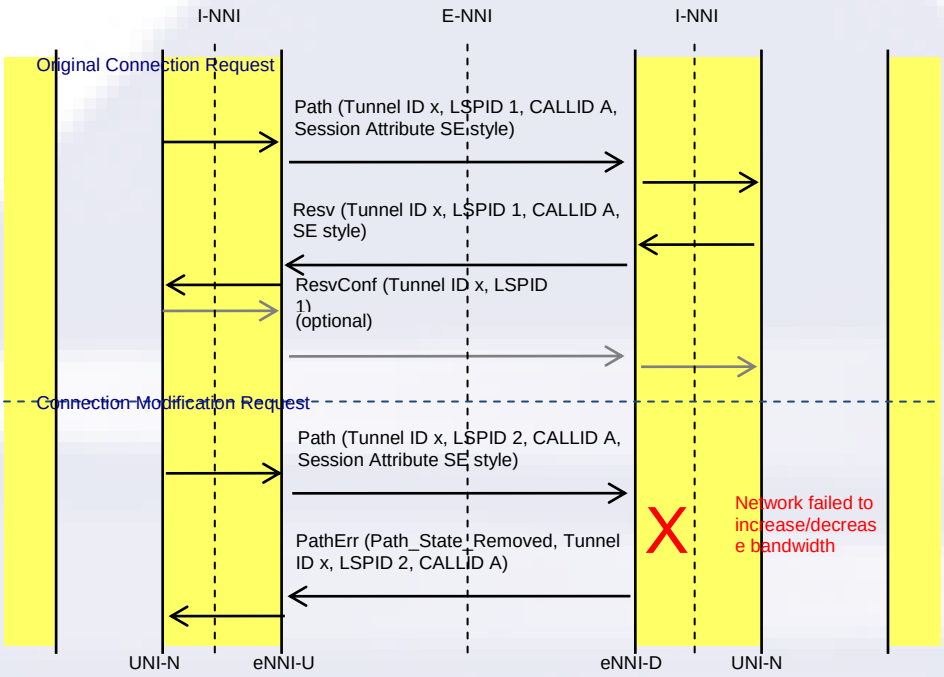


Figure 9: Connection Modification Failure

A bandwidth decrease can be achieved with an identical message flow, although the ResvConf message may not be necessary in this case. The new bandwidth becomes effective at the PathErr stage as opposed to the ResvConf message stage. A failure to decrease a connection bandwidth SHOULD result in a PathErr being sent for the Path message requesting less bandwidth. This MUST NOT impact the existing connections, other Path messages, or RSVP states.

6.4.2.3 VCAT Layer Call Modification Details

There are a few restrictions that apply to the modification of VCAT layer calls. The ordering of the members, i.e. server layer calls, MUST be preserved. When increasing VCAT layer bandwidth, new members MUST be added after existing members. For bandwidth decreases, any member may be removed but remaining members must maintain the original order. For simplicity, each call modification signaling sequence MUST be limited to either an increase or a decrease.

6.4.3 Connection Deletion

6.4.3.1 Graceful Connection Deletion Initiated from the Source or Destination

RSVP allows for deletion of connections using either a single pass PathTear message, or a ResvTear and PathTear message combination. Upon receipt of the PathTear message, a node deletes the connection state and forwards the message. In optical networks, however, it is possible that the deletion of a connection (e.g., removal of the cross-connect) in a node may cause the connection to be perceived as failed in downstream nodes (e.g., loss of frame, loss of light, etc.). This may in turn lead to management alarms and perhaps the triggering of restoration/protection for the connection.

To address this issue, the graceful connection deletion procedure MUST be followed. Under this procedure, an ADMIN_STATUS object with the D-bit set MUST be sent in a Path or Resv message along the connection's path to inform all nodes enroute of the intended deletion, prior to the actual deletion of the connection. The procedure is described in [RFC3473] and shown in Figure 10 and Figure 11.

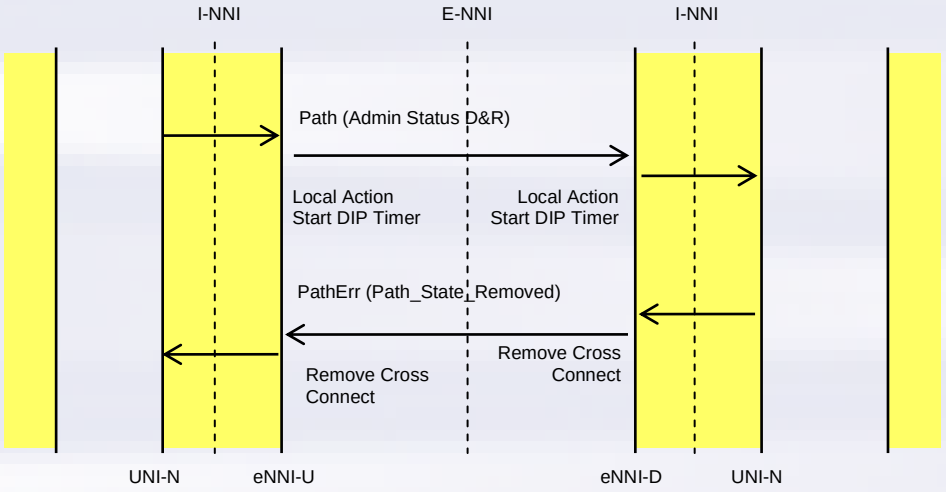


Figure 10: Connection Teardown Initiated by the Source

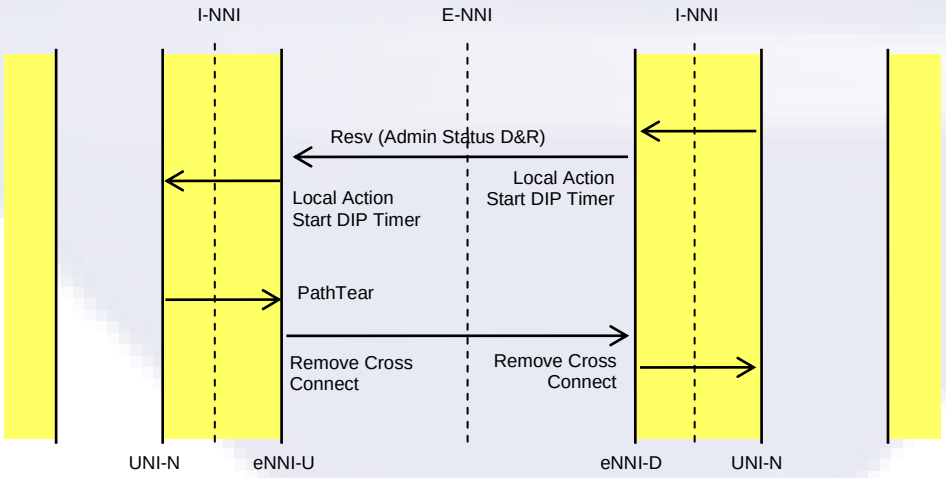


Figure 11: Connection Teardown Initiated by the Destination

6.4.3.2 Graceful Connection Deletion Initiated from the Network

A graceful deletion may also be initiated within the network. In this case, the eNNI-D may receive a connection deletion notification from the network. It is also possible that the eNNI-D can initiate a graceful deletion. In E-NNI Signaling 1.0, the network-initiated graceful deletion notification was signaled via the Path or Resv message with the A&R bits set. The signaling of graceful deletion is changed to use the Notify message in E-NNI Signaling 2.0 to align with [RFC3473].

6.4.3.2.1 Notify Message Support

The Notify message MUST be supported. In E-NNI Signaling, the Notify message is used to signal a connection deletion initiated from an E-NNI or network node. It is also used for recovery.

An eNNI-U node MUST include the NOTIFY_REQUEST object in the Path message sent to the eNNI-D. Likewise, an eNNI-D node MUST include the NOTIFY_REQUEST object in the Resv message to the eNNI-U. The Notify Node Address field of the NOTIFY_REQUEST object is set to the SC PC ID of the node generating the object.

When an E-NNI node needs to generate a Notify message, it targets the message to the SC PC ID associated with the Notify Node Address received in the incoming Path or Resv message. All session-specific objects SHALL be set to the appropriate values for the E-NNI connection segment.

An E-NNI node MUST NOT generate a Notify message to a signaling controller from which it did not receive a NOTIFY_REQUEST object. In addition, an E-NNI node SHOULD NOT generate a Notify message if it received a NOTIFY_REQUEST object but the Notify Node Address does not match its neighbor's SC PC ID.

6.4.3.2.2 Network Initiated Graceful Deletion

An E-NNI node MUST support the ability to forward a network initiated graceful deletion notification across the E-NNI interface. In addition, an E-NNI node MAY support the ability to initiate a graceful deletion notification. Network initiated graceful deletion is signaled across the E-NNI 2.0 interface using the Notify message instead of using the Path or Resv message with the ADMIN_STATUS A&R bits set.

An eNNI-D node initiates, or forwards, a graceful deletion notification by sending a Notify message to the eNNI-U. The graceful deletion Notify message contains the ADMIN_STATUS object with the D bit set. The eNNI-D also sends the graceful deletion Notify message when it receives a graceful deletion notification from the network.

In E-NNI Signaling 2.0, the graceful deletion notification SHOULD always be sent upstream to the source node. Upon receipt, the source node initiates the normal graceful deletion procedures as specified in [OIF-UNI-02.0]. The signal flow for a network initiated graceful deletion is shown in Figure 12.

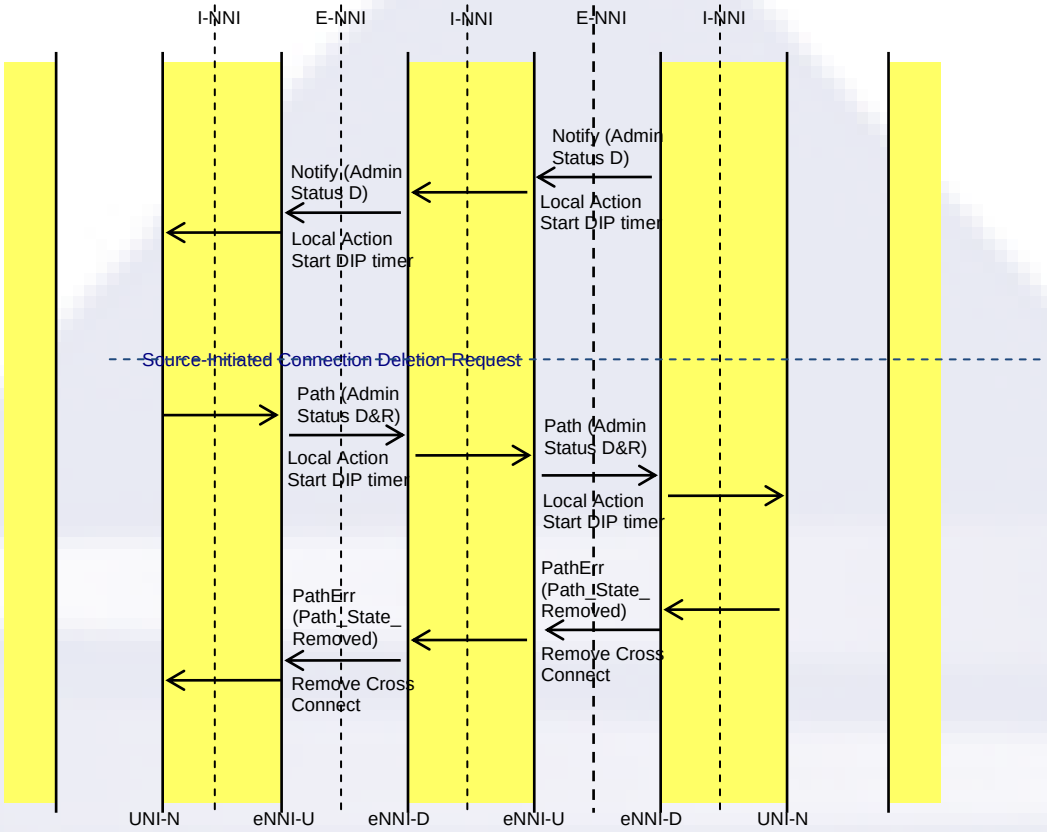


Figure 12: Connection Teardown Initiated by the eNNI-D

6.4.3.2.3 E-NNI Signaling 1.0 Compatibility

An E-NNI 2.0 node MUST NOT send the Notify message for a graceful deletion notification if its neighbor only supports E-NNI Signaling 1.0. Instead, the E-NNI 2.0 node MUST send a Path or Resv message containing the ADMIN_STATUS object with the A&R bits set to signal network graceful deletion.

An E-NNI 2.0 node determines its neighbor's E-NNI version support either by manual configuration or through an automatic discovery process. An E-NNI node MAY assume the neighbor is running E-NNI 1.0 Signaling if it does not receive a NOTIFY_REQUEST object or if the NOTIFY_REQUEST object is received but the Notify Node Address is not equal to the neighbor's SC PC ID.

E-NNI Signaling 1.0 also allows for graceful deletion notifications in the downstream direction. In this case, the eNNI-U SHOULD signal the downstream graceful deletion to an E-NNI 2.0 compliant eNNI-D by using the Notify message containing the ADMIN_STATUS object with the D bit set. Otherwise, it SHOULD send a Path message with the A&R bits set in the ADMIN_STATUS object if the eNNI-D is E-NNI 1.0 compliant.

6.4.3.3 Forced Deletion

An E-NNI node SHOULD support the ability to initiate a forced deletion of a connection. A forced deletion may be necessary to react to events such as:

- Internal network failures, which force the network to terminate connections
- When the "Deletion In Progress" timer object expires

An eNNI-U node initiates a forced deletion by deleting its RSVP states and removing the cross connect. It then sends a PathTear message downstream to the eNNI-D while at the same time signaling a forced deletion in the upstream direction. The eNNI-D, upon receipt of the PathTear message, deletes its RSVP states and removes the cross connect. The eNNI-D continues the signaling of the forced deletion in the downstream direction. This signal flow is shown in Figure 13.

Figure 14 shows a forced deletion initiated by an eNNI-D node. In this case, the eNNI-D deletes its RSVP states and removes the cross connect. The eNNI-D then signals the forced deletion by sending a PathErr message with the "Path_State_Removed" flag set to the eNNI-U and simultaneously signals a forced deletion in the downstream direction. The eNNI-U will delete its RSVP states and remove the cross connect when it receives the PathErr message. The eNNI-U will continue to propagate the forced deletion signal upstream through the network.

It is also possible that the network may generate a forced deletion signal. When an eNNI-U receives the forced deletion signal from the upstream network, it deletes the RSVP states, removes the cross connect, and signals PathTear to the eNNI-D. Likewise, if an E-NNI-D receives a forced deletion signal from the downstream network, it deletes the RSVP states, removes the cross connect, and signals PathErr with the "Path_State_Removed" flag set to the eNNI-U.

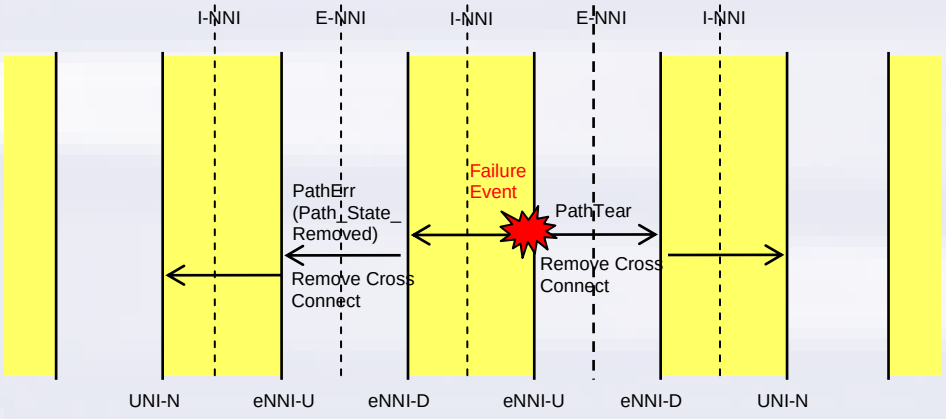


Figure 13: Forced Deletion Initiated by an eNNI-U

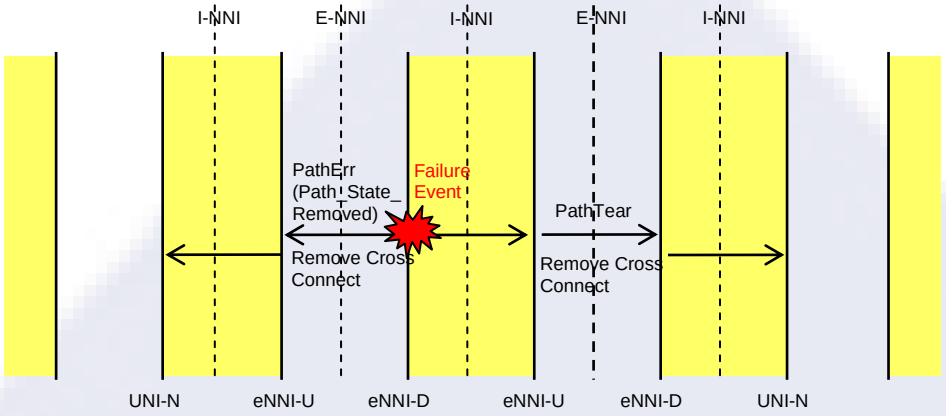


Figure 14: Forced Deletion Initiated by an eNNI-D

6.4.4 Additional RSVP-TE Messages

In addition to the signal flows described in [OIF-ENNI2.0-SIG] Section 12, RSVP-TE provides the ACK message. This message is only transmitted between eNNI-U and eNNI-D and may be used:

- To obtain an acknowledgement for sent messages. The acknowledgement function can be provided either directly, using the Ack message, or indirectly (via MESSAGE_ID_ACK) when the sent message has a corresponding reply message (that is immediately generated) on a specific link (e.g., Resv/PathErr is Path's corresponding reply message). Figure 15 illustrates an additional ACK for the case of connection setup.

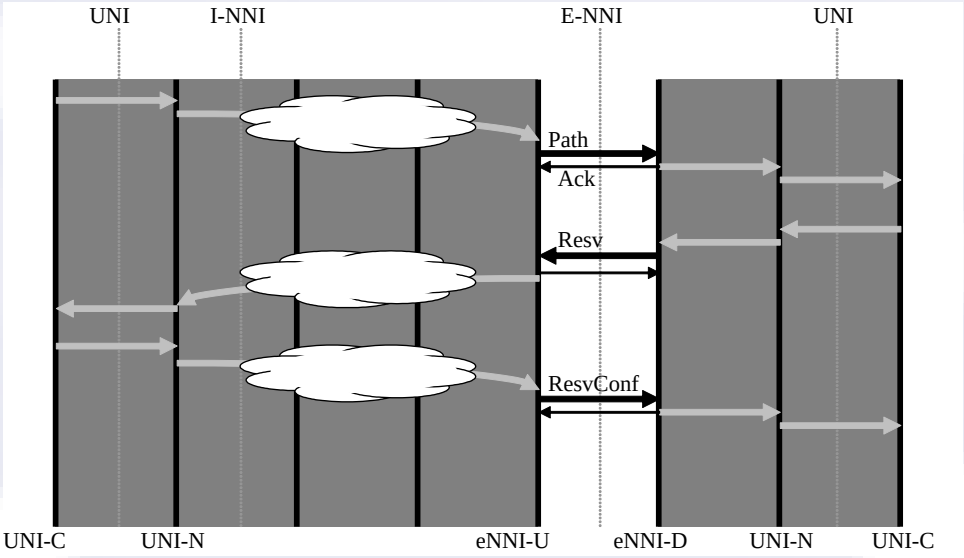


Figure 15: Basic SC Setup Using RSVP-TE

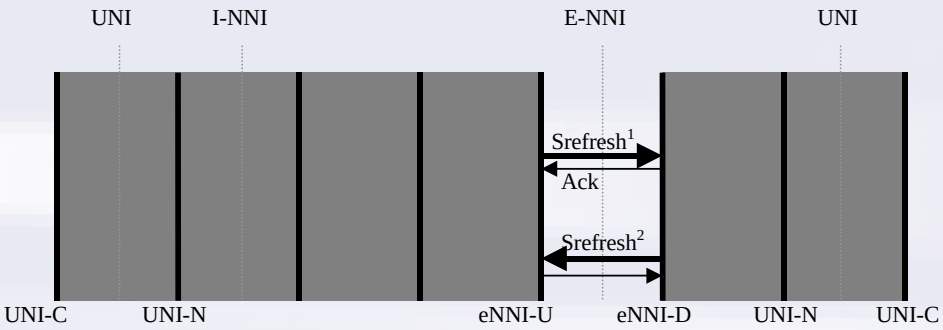
RSVP-TE also defines Hello and Srefresh messages. Both messages have local scope and are specific to the RSVP-TE protocol. The Hello message may be used:

- to ensure an RSVP session is up (using request and acknowledge objects)

- to initiate restart procedures by exchanging recovery and restart timers

The Srefresh message may be used:

- to refresh RSVP-TE state without the transmission of Path or Resv messages. This reduces the amount of information that must be transmitted and processed to maintain connection state synchronization. A Srefresh message carries a list of Message_Identifier fields corresponding to the Path and Resv trigger messages that established the state. Message_Identifier fields are carried in a MESSAGE_ID_OBJECT. Figure 16 illustrates an example of Srefresh used to refresh Path and Resv states.



Note 1: This Srefresh may be used to refresh both Path and Resv state information associated with all connections from eNNI-U to eNNI-D.
Note 2: This Srefresh may be used to refresh both Path and Resv state information associated with all connections from eNNI-D to eNNI-U.

Figure 16: Basic Srefresh Signaling

6.5 RSVP-TE Control Plane Failures

6.5.1 RSVP-TE Signaling Channel Failure

As described in [OIF-ENNI2.0-SIG] Section 12.2.4, the failure of a signaling channel or control protocol entities MUST NOT result in the deletion of previously established connections. The handling of control state failure (without loss of the forwarding state) is described in [RFC3473] through the support of the RESTART_CAP object[5], which requires the use of Hello messages. Here, in particular, a node MUST support the fault handling procedure described in Section 9 of [RFC3473].

In addition to the behaviors described in [OIF-ENNI2.0-SIG] Section 12.2.4, RSVP-TE requires an exchange of messages to synchronize the states of established connections. During a signaling channel failure, a self-refresh procedure is executed to prevent state information from expiring. After recovery from the failure, the neighboring control entities initiate an exchange of Hello messages. The Hello messages are used to trigger the process of synchronizing (or recovering)

[5] To support the requirement that a control plane failure does not affect established connections, the Restart Time used in the RESTART_CAP object of the Hello message MUST be set to 0xffffffff. Note that local policy or configuration rules that are set based on management input may override the values specified by the Restart Time.

the states of established connections. This ensures that the states of established connections remain consistent. The following local behaviors apply to nodes impacted by the signaling channel failure:

- A control plane node detecting a signaling channel failure should inform the management system of the failure. The default (control plane) behavior is to enter self-refresh of the call/connection states. The management system may give the control plane specific instructions to override the default behavior, for example, to release certain connections. As an example, possible management system instructions may be to remain in self-refresh mode, or to release certain connections.
- A control plane node (NCC or CC) detecting that one (or more) connections cannot be synchronized with its neighbor (e.g., due to different states for the call or connection) should inform the management system. The default behavior of the control plane should be to retain the connection unless explicitly instructed to release the connection by an external entity. As an example, possible management system instructions may be to delete the connection. Specifics of the interactions between the control plane and management plane are outside the scope of this document.

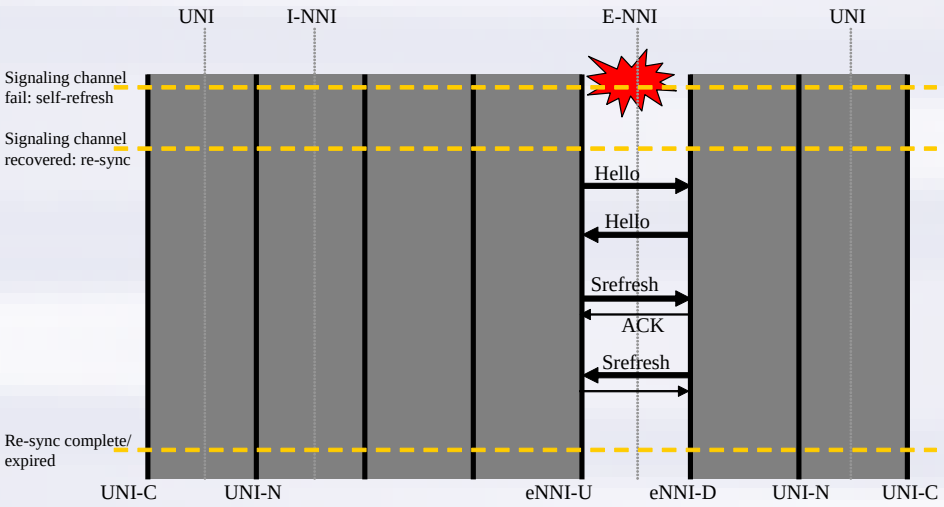


Figure 17: Recovery from Signaling Channel Failure

To ensure RSVP Hellos are supported, a node MUST NOT accept any call request unless a successful control adjacency has been established. A PATH message received from a node that has not sent any Hello message yet MAY be used to trigger the Hello procedure. A node SHOULD respond to unexpected or erroneous Hello messages by setting the Dst_Instance to 0 in the Hello Request or HelloAck object, which indicates that the received message is not accepted.

6.5.2 RSVP-TE Control Plane Failure

In addition to the behaviors described in [OIF-ENNI2.0-SIG] Section 12.2.4, RSVP-TE requires a message exchange to synchronize the states of established connections. During a control plane node (CC) failure, a self-refresh procedure is executed to prevent state information from expiring. After recovery from the failure, the recovered node (CC) must attempt to restore the state information of established connections from its local persistent storage [G.7713.2]. Subsequent to this, the neighboring control entities initiate exchanges of Hello messages. The Hello messages are used to trigger synchronizing (or recovering) the states of established connections. This

ensures that the states of established connections remain consistent. Thus the following local behaviors can be envisioned for handling control plane node (CC) failure:

- A control plane node (NCC/CC) must provide for persistent storage of call and connection state information. This allows each control plane node (NCC/CC) to recover the states of calls or connections after recovery from a signaling controller entity failure or reboot (or loss of local state in memory). Note that although the restart mechanism allows neighboring control plane nodes (NCCs/CCs) to recover (and thus infer) the states of calls or connections automatically, this mechanism can be used to verify neighbors' states while the persistent storage provides the local recovery of lost state. In this case, per [RFC3473], if during the Hello synchronization the restarting node (NCC/CC) determines that a neighbor does not support state recovery, and the restarting node (NCC/CC) maintains its state on a per neighbor basis, the restarting node (NCC/CC) should immediately consider the Recovery to be complete.
- A control plane node (NCC/CC) detecting that one (or more) connections cannot be synchronized with its neighbor (e.g., due to different states for the call or connection) should inform the management system. The default behavior of the control plane should be to retain the connection unless explicitly instructed to release the connection by an external entity. The management system may give the control plane further instructions on how to handle the non-synchronized connection. As an example, possible management system instructions may be to delete the connection. Specifics of the interactions between the control plane and management plane are outside the scope of this document.
- A control plane node (NCC/CC), after recovering from node failure, may not be able to recover neighbor forwarding adjacency information from its local persistent storage and thus may lose information on forwarding adjacencies. In this case the control plane node (NCC/CC) should query an external controller (e.g., the management system) for information to recover the forwarding adjacency information. Specifics of the interactions between the control plane and management plane are outside the scope of this document.

6.6 Security Note

Note that using the security attribute defined in [G.7713] or the RSVP INTEGRITY object described in [RFC2747] (which is updated by [RFC3097]) for securing the OIF Control Plane is NOT RECOMMENDED because the [G.7713] security attribute is not specifically defined, and the RSVP INTEGRITY object :

- only covers one protocol. A single security solution for all Control Plane protocols is desired.
- does not provide the required confidentiality service or any automated method for exchanging and updating keys.
- specifies MD5 as its only security transform, and MD5, as a hash function, is now considered a weak mechanism.

The rationale given in [RFC2747] for rejecting IPsec does not apply to RSVP-TE as used in the OIF Control Plane.

6.7 Call/Connection Recovery

This amendment proposes re-using the PROTECTION and the ASSOCIATION objects (defined in [RFC4872] and [RFC6780]) to specify respectively the recovery type information and the association information.

OIF RSVP E-NNI signaling is used to convey these two objects between recovery domains and, in the case of nested recovery domains, within the larger recovery domain between nested recovery domains.

6.7.1 Recovery mechanism-independent signaling

6.7.1.1 Source and destination UNI-Ns belong to the recovery domain

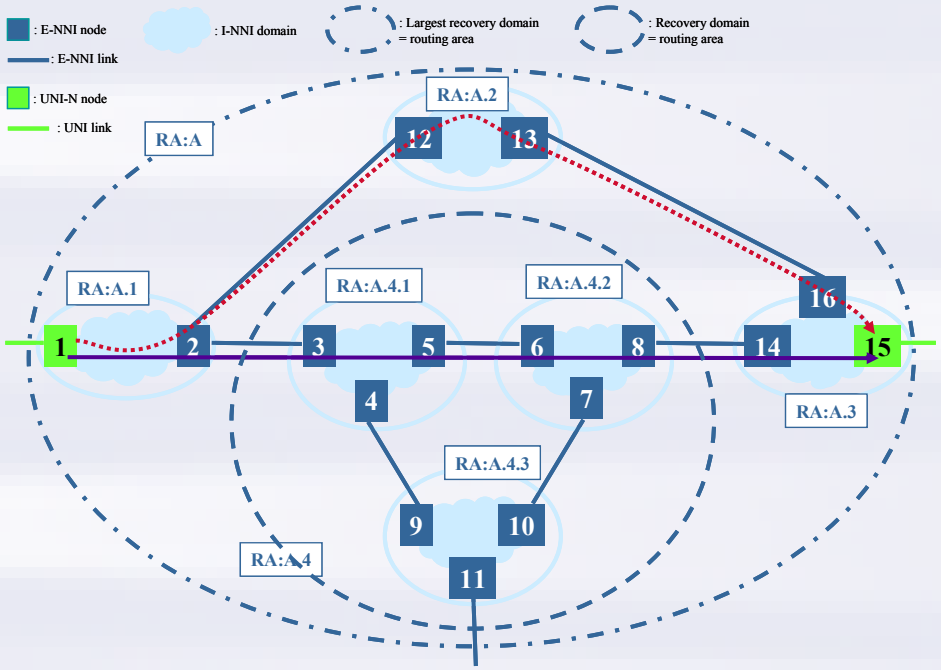


Figure 18: Example of recovery connection between source and destination UNI-Ns

When both the source and destination UNI-Ns belong to the same recovery domain[6], the source UNI-N may choose a recovery mechanism for the new call.

If the UNI-N node has received an UNI request, it should look at the G-UNI Service Level sub-object and refer to its local policy to map the service level to one (or several) recovery mechanism(s).

This mechanism is considered an end-to-end recovery mechanism.

[6] This may not be the case when the two UNI-Ns belong to different carriers that have no common recovery service level agreement (in such a case, the largest recovery domains would be contained uniquely within each carrier network).

The Path message for the working connection will carry an ASSOCIATION object and a PROTECTION object. Both objects refer to whatever recovery mechanism has been selected by the source UNI-N.

The Path message for the recovery connection will carry an ASSOCIATION object and a PROTECTION object too. The ASSOCIATION object specifies exactly the same values as the ASSOCIATION object used in the working connection Path message.

In both cases, the ASSOCIATION and PROTECTION objects are encapsulated in an E-NNI recovery sub-object contained in the OIF_RECOVERY_STACK object.

When the working or recovery connections Path message enters nested recovery domains, these two (end-to-end) ASSOCIATION and PROTECTION objects are carried unchanged within and across those domains.

Section 6.7.1.3 then applies for subsequent messages processing.

6.7.1.2 Signaling in the largest recovery domain

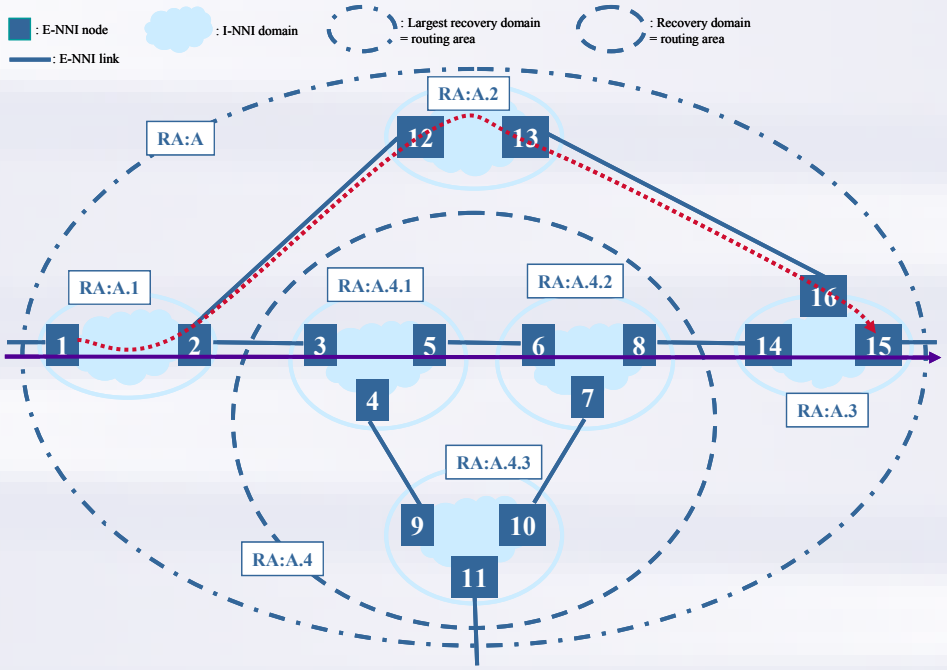


Figure 19: Example of recovery connections in largest recovery domain

When the DIN from the largest recovery domain receives an E-NNI connection request (Path message) from outside that domain:

- It MUST discard and ignore, if any, all set of recovery information received (PROTECTION, ASSOCIATION, OIF_RECOVERY_STACK …objects);
- It should look at the G-UNI Service Level sub-object and refer to its local policy to map the service level to one (or several) recovery mechanism(s).
- Add an ASSOCIATION object and a PROTECTION object as per section 6.7.1.1. (first E-NNI Recovery sub-object in the OIF_RECOVERY_STACK object).

Section 6.7.1.3 then applies for subsequent messages processing.

6.7.1.3 Signaling in a nested recovery domain

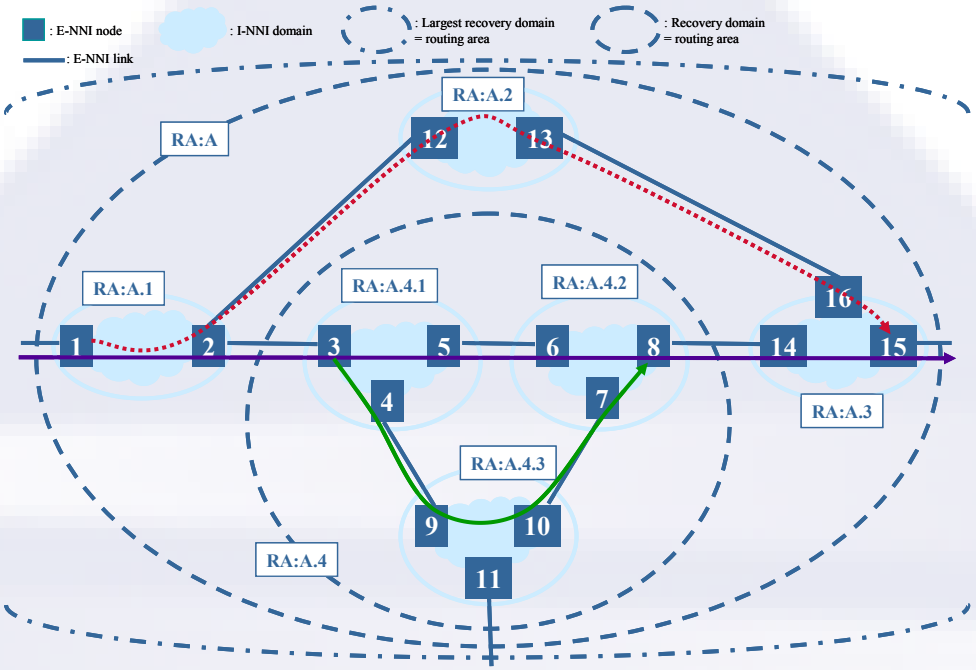


Figure 20: Example of recovery connections in nested domains

6.7.1.3.1 Path message processing - DIN

When a DIN receives a Path message, it checks whether an OIF_RECOVERY_STACK object is present.

- If there is none, it looks at the G-UNI Service Level sub-object and refers to its local policy to map the service level to one recovery mechanism, or none.

  If a recovery mechanism has been selected, it adds an ASSOCIATION object and a PROTECTION object as per section 6.7.1.1. (first E-NNI Recovery sub-object in the OIF_RECOVERY_STACK object).

- If there is one, it looks at the first E-NNI Recovery sub-object (the latest sub-object added in the stack) domain local service level and refer to its local policy to map the domain local service level to one (or several) recovery mechanism(s), or none.

  If a recovery mechanism has been selected, it adds an ASSOCIATION object and a PROTECTION object in a new E-NNI Recovery sub-object. Its domain local service level is set appropriately. This sub-object is pushed in the OIF_RECOVERY_STACK object (i.e., it becomes the first E-NNI Recovery sub-object, see section 6.3.11.3).

Note that if a node is a DIN for multiple nested recovery domains, and one recovery mechanism is selected in each domain, he may push multiple E-NNI Recovery sub-objects in the OIF_RECOVERY_STACK object.

6.7.1.3.2 Path message processing - DEN

When a DEN receives a Path message, it checks whether an OIF_RECOVERY_STACK object is present, and whether the first E-NNI Recovery sub-object identifies its local recovery domain.

- If such E-NNI Recovery sub-object is found, it pops this sub-object from the OIF_RECOVERY_STACK object.

Note that if a node is a DEN for multiple nested recovery domains, he may pop multiple E-NNI Recovery sub-objects from the OIF_RECOVERY_STACK object.

6.7.1.3.3 Resv message processing - DEN

When a DEN receives a Resv message for a connection for which a recovery mechanism was selected in its local domain, it pushes a new E-NNI Recovery sub-object in the OIF_RECOVERY_STACK object (i.e., it becomes the first E-NNI Recovery sub-object, see section 6.3.11.3).

Note that if a node is a DEN for multiple nested recovery domains, he may push multiple E-NNI Recovery sub-objects in the OIF_RECOVERY_STACK object.

6.7.1.3.4 Resv message processing - DIN

When a DIN receives a Resv message, it checks whether an OIF_RECOVERY_STACK object is present, and whether the first E-NNI Recovery sub-object identifies its local recovery domain.

- If such E-NNI Recovery sub-object is found, it pops this sub-object from the OIF_RECOVERY_STACK object.

Note that if a node is a DIN for multiple nested recovery domains, he may pop multiple E-NNI Recovery sub-objects from the OIF_RECOVERY_STACK object.

6.7.2 Connections association for recovery purpose

[RFC6780] ASSOCIATION object allows associating LSPs belonging to different RSVP-TE sessions based on its association type, its association source and its association identifier.

In ASON networks, an ASSOCIATION object will be carried across E-NNI interfaces through multiple domains. Consequently:

- The association source and identifier must be independent from the RSVP SESSION and SENDER_TEMPLATE object fields, since they change across each E-NNI (using the LSPID as in [RFC6780] is not possible);

When a Path message is received for the recovery connection, CCs inside of the recovery domain need to be able to identify if they are DEN of the recovery domain, and should act accordingly. If they are not, then the signaling should progress through the CC towards the actual DEN.

In case of restoration, it is possible the network topology causes a restoration path to intersect the original path at a point other than the restoration domain border egress node. An example network topology is shown in Figure 21, where Node B is involved in the original path (A-B-Z) as well as the restoration path (A-E-B-F-Z). In this case, when the signaling for the restoration path enters node B, it should continue on to the domain egress node, Node Z, through Node F, without any special operation (note that in Figure 21 example, the DIN - Node A – chose not to reuse any resource upstream or downstream to Node B).

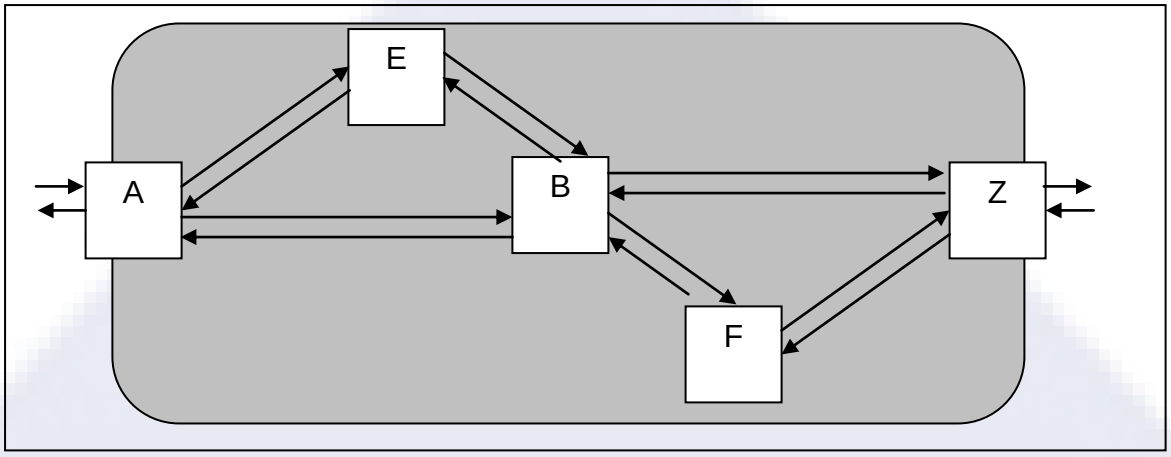


Figure 21: Original and restoration path cross prior to domain egress

For recovery domains scoped to a routing area, a node will identify that it is the recovery domain DEN of a working or recovery connection if the connection downstream link leaves the recovery domain (see also section 6.7.2.1 that requires the specification of the E-NNI exit link in a recovery connection Explicit Route).

6.7.2.1 Recovery connection endpoint for a transit recovery domain

This section does not apply to E-NNI-scoped recovery domains.

The working and recovery connections of a transit recovery domain must exit this transit recovery domain (i.e. a domain where the destination TNA is not located) through the same physical node. Therefore, the recovery connection Explicit Route MUST specify the transit domain exit E-NNI link. In case of abstraction, the recovery connection will then be routed to the same physical exit node as the working connection, as shown in Figure 22.

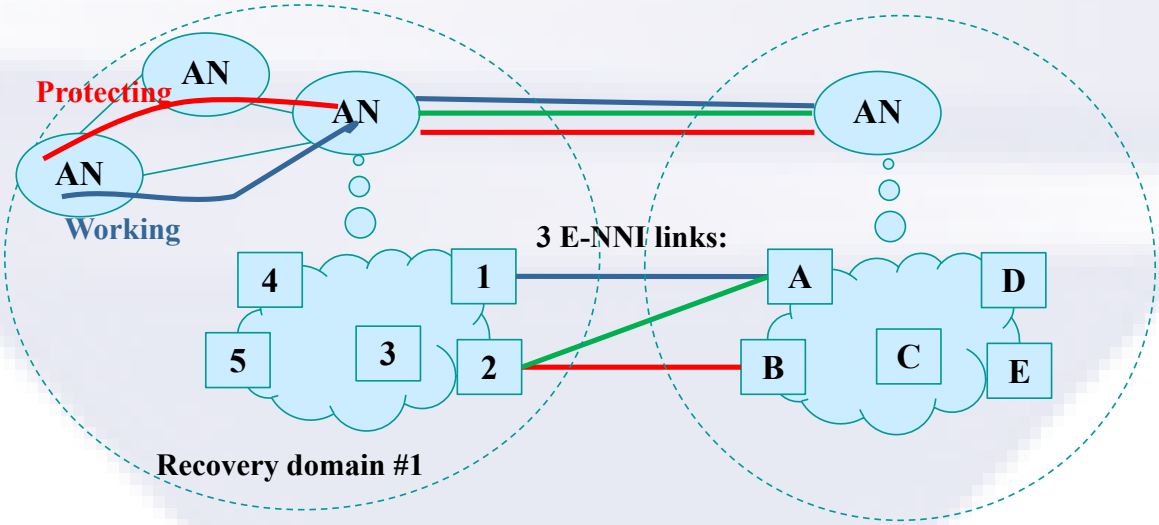


Figure 22: Recovery connection endpoint (transit recovery domain)

6.7.2.2 Recovery connection endpoint for the destination recovery domain

In the destination recovery domain, the working and recovery connections must terminate on the same physical node, which is identified by the destination TNA. In case of abstraction, as shown in Figure 23 for instance, no ERO is specified in the Path message received by the signaling controller in charge of the final abstract node (AN8), the recovery connection is then be routed thanks to the destination TNA.

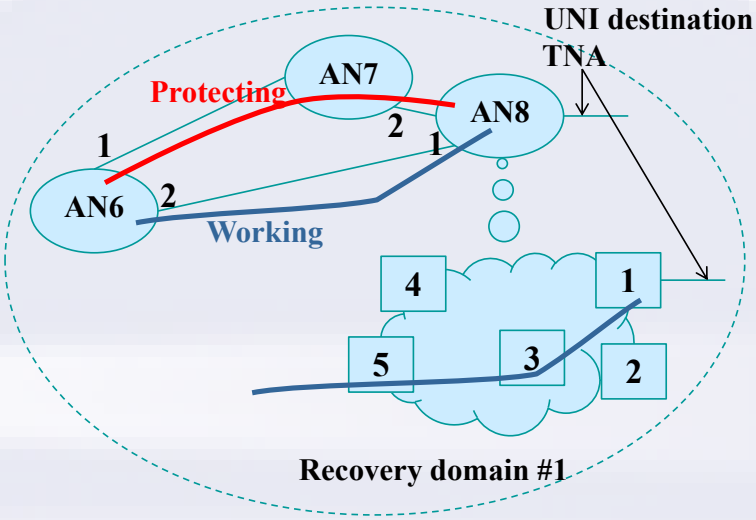


Figure 23: Recovery connection endpoint (destination recovery domain)

6.7.3 Resource re-use

Resource re-use across an E-NNI link can be achieved by using RSVP SE reservation style and having the recovery connection using the same Tunnel ID as the working connection. This mechanism does not allow resource re-use within I-NNI domains though (for instance within the recovery domain A.4.2 in Figure 24).

If resource re-use is desired within I-NNI domains, then an additional ASSOCIATION object should be added as follow:

- Association Type: Resource Sharing ([RFC4873]);
- Association source and identifier: set as specified in section 6.7.2.
- For hard-rerouting, if resource re-use is desired, the Resource Sharing ASSOCIATION object must be added when the working connection is established (after a failure occurs, it may be impossible to add this object to the working connection – to allow resource re-use with the recovery connection that will be established - if the control plane is impacted by the failure too);
- For soft-rerouting, if resource re-use is desired, and if no Resource Sharing ASSOCIATION object was specified during the establishment of the connection being soft-rerouted, this object may be added to that connection at the time the soft-rerouting is triggered (and of course the same object must be added during the establishment of the new rerouting connection that follows).

In Figure 24, the source node [1] initiates the hard rerouting after it get the failure notification specifying E-NNI link [5-6], however the SNC[7] switches will be implemented E.g. in nodes [3] and [8] [8]instead of at the end-points: nodes [1] and [15] in a more classical approach.

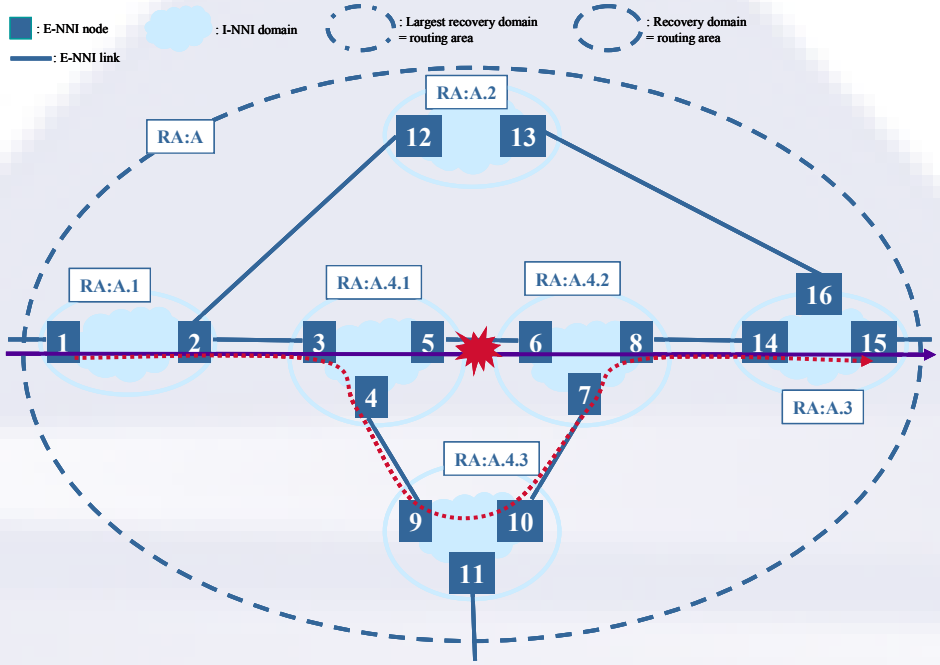


Figure 24: Example of re-use of resources over E-NNI links [2-3] & [8-14]

6.7.4 Failure notification

6.7.4.1 Failure specification

A PathErr message is sent to report the detected failure to an upstream DIN or DEN. The Path_State_Removed Flag of the ERROR_SPEC object must not be set.

A Notify message may be sent in addition. Its receipt must not abort the recovery process.

An IF_ID ERROR_SPEC object is used for the purpose of identifying the failure using two TLVs: an IF_INDEX TLV and a REPORTING_OSPF_AREA TLV.

> The REPORTING_OSPF_AREA identifies the recovery domain/routing area into which the failure identification (IF_INDEX TLV fields) can be interpreted. Indeed, lower level recovery domains may be involved in the signaling involved in an upper level's recovery domain. Therefore, the lower level area has the potential to impact the information carried in the signaling messages. These areas must be able to identify if the Notify message being processed is within the scope of the lower level recovery domain and ignore Notify messages that are not within the scope of the lower level recovery domain and pass through the IF_ID ERROR_SPEC received from outside the area untouched.

[7] Sub-Network Connection

[8] Note that other nodes inside RA:A.4.1 and RA.A.4.2 can implement the SNC switches. This decision is left to the DIN of these domains (nodes 3 and 7) but is out of scope of this amendment.

When a Notify message is received by the ingress node to a domain, it will look to see if the REPORTING_OSPF_AREA TLV matches the routing area of the downstream interface the Notify message was received on.

In the Abstract Node case described in E-NNI Recovery amendment, an LINK_EXCLUSIONS TLV containing a list of ingress links to the abstract node's border node which should be avoided may also be included in the IF_ID ERROR_SPEC.

Per [RFC4872][9], the error code and error value are set to "Notify Error (25) / LSP Locally Failed (11)" when a failure is detected. The error code and error value are set to "Notify Error (25) / LSP Recovered (10)" when a failure is cleared. The error node is originally set to the SC ID of the signaling controller having detected the failure. When a PathErr message being forwarded leaves a (N) recovery domain and enters an (N+1) upper recovery domain, the error node must be translated to the (N) DIN SC ID.

The IF_ID ERROR_SPEC TLVs are processed as follows:

- When a transit node within a recovery domain, or the DEN of a recovery domain, detects a failure, it generates a PathErr message whose IF_INDEX TLV specifies a transport node identifier and a link identifier. The REPORTING_OSPF_AREA TLV identifies that recovery domain. As long as this PathErr message is forwarded within the recovery domain where the failure occurred, these TLVs are not modified;
- When the DIN of a recovery domain detects a failure, it generates a PathErr message whose IF_INDEX TLV specifies identifiers (abstract node or abstract link identifiers) that belong to the upper recovery domain; The REPORTING_OSPF_AREA TLV identifies that upper recovery domain.
- When the DIN of a recovery domain forwards a PathErr message outside of its recovery domain and into an upper recovery domain, it translates the IF_INDEX TLV identifiers into identifiers (abstract node or abstract link identifiers) that belong to the upper recovery domain. The REPORTING_OSPF_AREA TLV is translated too to identify the upper recovery domain.

This error node and TLVs translations are performed by the DIN of a recovery domain being exited and where the failure could not be recovered. This translation does not occur when the PathErr message is forwarded within the same recovery domain the REPORTING_OSPF_AREA TLV is referring to.

For instance, in Figure 25 below, node 7 will send to node 10 a PathErr message specifying an IF_ID TLV within the scope of RA A.4 (e.g., abstract node A.4.2). This TLV will not be modified by node 9 when it forwards the PathErr message to node 4. Only node 3 will modify the IF_ID TLV when it forwards the PathErr message to node 2 (assuming here the connection cannot be restored within A.4), specifying information in the scope of RA A (e.g.., abstract node A.4).

[9] Note that [RFC4872] "Notify Error (25) / LSP Failure (9)" is not used in this amendment, since the switchover coordination is not handled by the control plane.

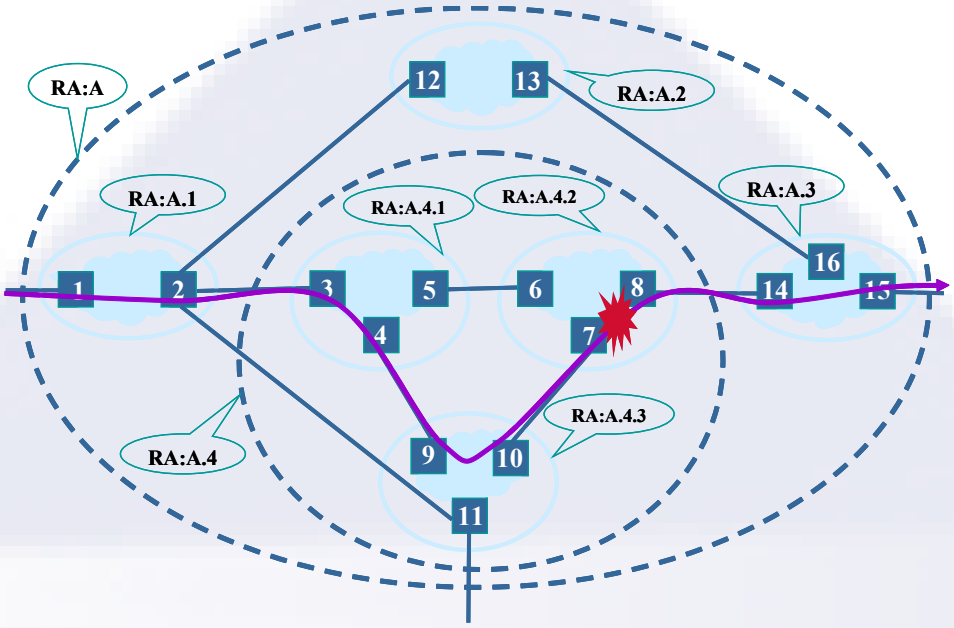


Figure 25: PathErr message forwarding

6.7.4.2 PathErr message forwarding

The PathErr message identifying a transport plane failure must be forwarded "hop-by-hop" across E-NNI interfaces, toward the source call controller.

Every time a domain is traversed, the identifiers of the connection received at the upstream/downstream E-NNI will have to be translated to the identifier used in the downstream/upstream E-NNI.

6.7.4.3 Notify message forwarding

Usage of a Notify message is optional. Its forwarding follows the same rules described in the previous sections. The same error codes and error values specified in section 6.7.4.1are used.

A Notify message is sent to the CC whose SC ID was specified in the Path message.

Every time a domain is traversed, the identifiers of the connections listed in the <notify session list> (<upstream notify session>) object received at the upstream/downstream E-NNI will have to be respectively translated to the identifiers used in the downstream/upstream E-NNI.

Note also that when a CC receives a single Notify message listing multiple connections, because the upstream CCs for such connections may not be the same, the CC may forward upstream multiple Notify messages (each one toward different upstream CCs).

> For instance, in Figure 26 below, Node 14 may send a single Notify message identifying both the red and purple connections, assuming both being impacted by the failure in A.3. When the Notify message reaches node 6, this node will have to send two Notify messages, one to node 5 and another to node 10.

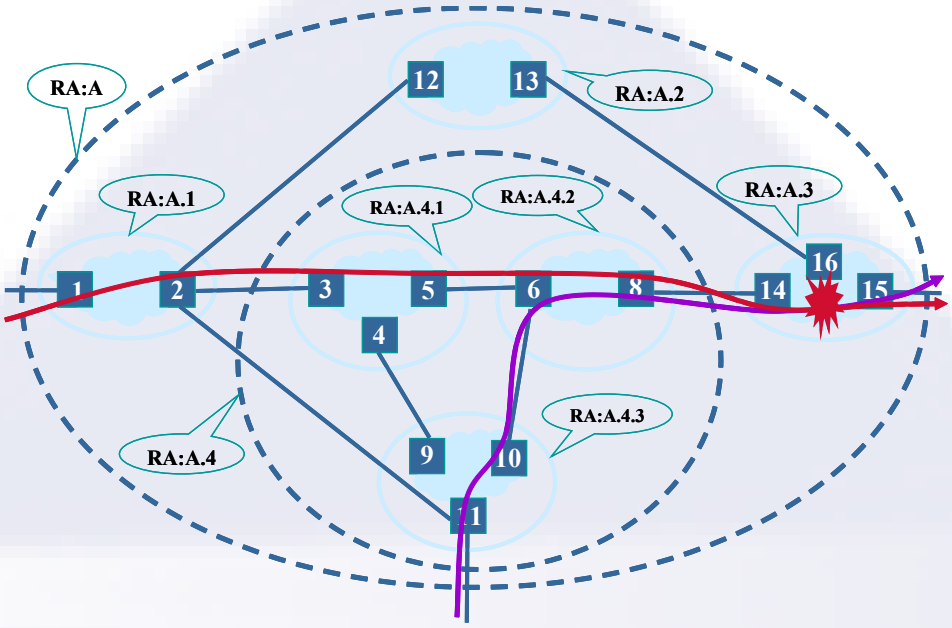


Figure 26: Example of forwarding a failure notification referencing multiple connections

6.7.5 Recovery mechanism-dependent signaling procedures

6.7.5.1 1+1 Protection

As shown in Figure 27, the PROTECTION object is set as follows:

LSP (Protection Type) Flags: 0x08 (1+1 Unidirectional Protection) or 0x10 (1+1 Bidirectional Protection)

Secondary (S) / Protecting (P) / Notification (N) / Operational (O) bits:

- SPNO=0010b for the working connection setup;
- SPNO=0110b for the recovery connection setup;
- SPNO=0111b for the recovery connection activation.

The ASSOCIATION object is set as specified in section 6.3.11.3.2, the Association Type being set to 0x01 (Recovery).

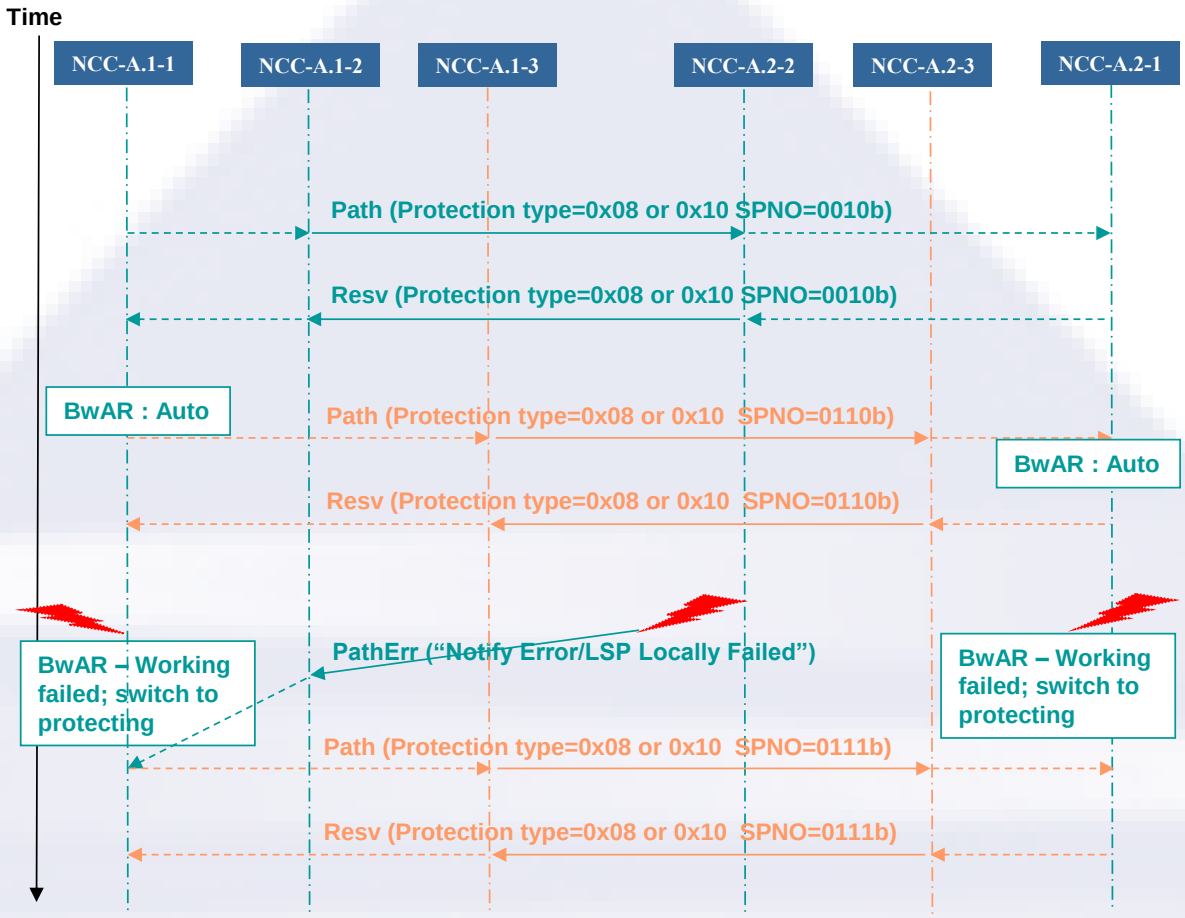


Figure 27: RSVP-TE Signaling for 1+1 Protection

Figure 27 is based on [OIF-ENNI-REC-AM-01.0] Figure 3 reference network. "BwAR", "B&R" and "XC" are terms defined in [OIF-ENNI-REC-AM-01.0].
Figure 27 assumes a bi-directional failure. In case of uni-directional failure, the automatic roll kicks in on one side only.

6.7.5.2 Shared-mesh Restoration

As shown in Figure 28, the PROTECTION object is set as follows:

LSP (Protection Type) Flags: 0x02 Rerouting without Extra-Traffic

Secondary (S) / Protecting (P) / Notification (N) / Operational (O) bits:

- SPNO=0000b for the working connection setup;
- SPNO=1100b for the recovery connection setup;
- SPNO=0100b for the recovery connection activation.
- SPNO=1100b for the recovery connection de-activation (which implicitly activates the working connection – see the abstract message sequence diagram in the E-NNI Recovery amendment).

The ASSOCIATION object is set as specified in section 6.3.11.3.2, the Association Type being set to 0x01 (Recovery).

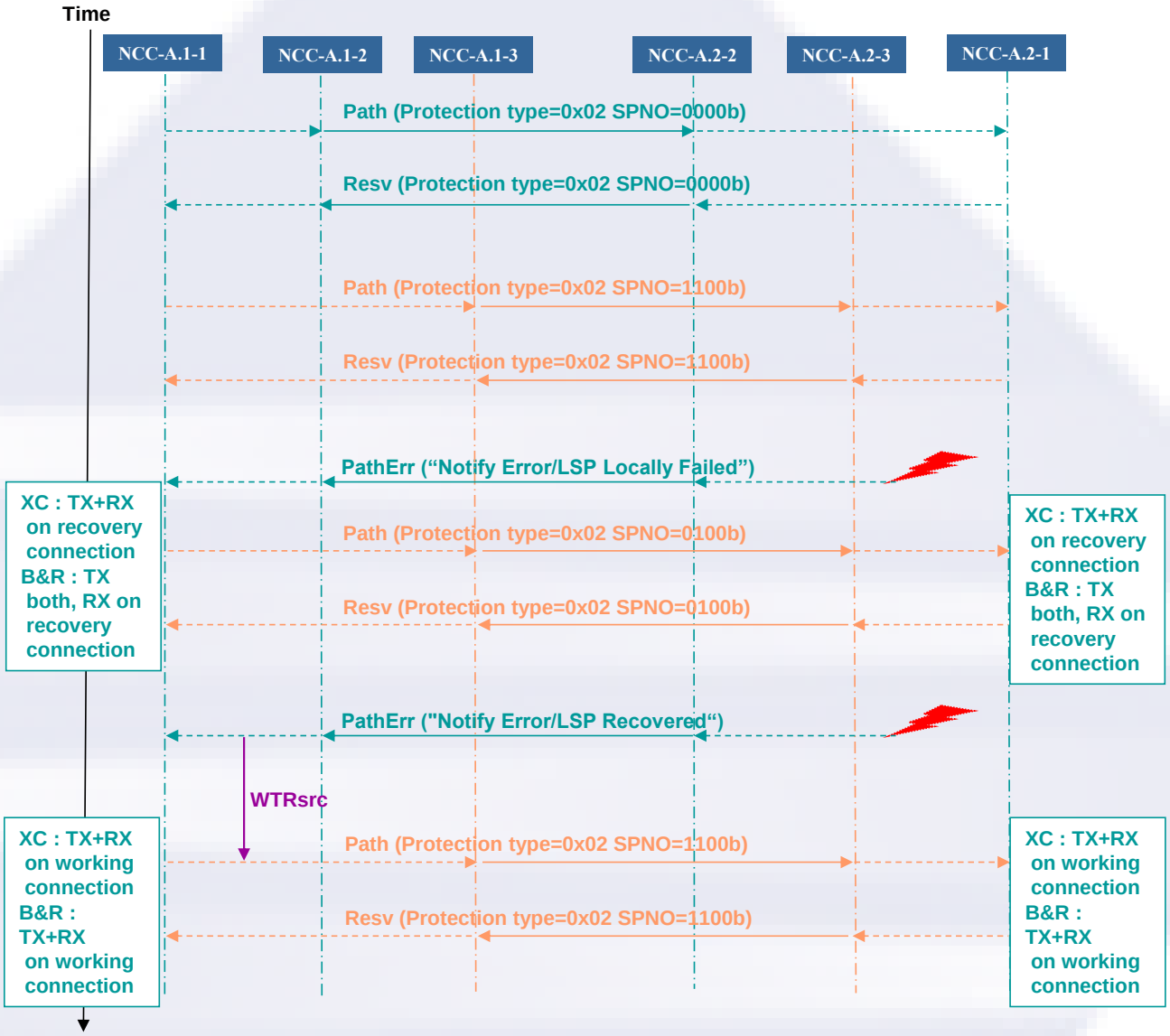


Figure 28: RSVP-TE Signaling for shared-mesh restoration recovery

Figure 28 is based on [OIF-ENNI-REC-AM-01.0] Figure 3 reference network. "BwAR", "B&R" and "XC" are terms defined in [OIF-ENNI-REC-AM-01.0].

6.7.5.3 Revertive full-rerouting

As shown in Figure 29, the PROTECTION object is set as follows:

LSP (Protection Type) Flags: 0x01 (Full) Rerouting

Secondary (S) / Protecting (P) / Notification (N) / Operational (O) bits:

- SPNO=0000b for the working connection setup;

- SPNO=0000b for the combined recovery connection setup and activation;
- SPNO=1000b for the recovery connection de-activation (which implicitly activates the working connection – see the abstract message sequence diagram in the E-NNI Recovery amendment).

The ASSOCIATION object is set as specified in section 6.3.11.3.2, the Association Type being set to 0x01 (Recovery). A second ASSOCIATION object can be used with the Association Type set to Resource Sharing if resource reuse is desired as specified in section 6.7.3.

It is a local policy at the DEN to determine when the B&R mechanism is established[10].

[10] The DEN controller does not know at the time the recovery connection is setup, whether the full-rerouting operation is revertive or non-revertive (see also section 7.7.5.4). Therefore, if B&R is supported, the DEN controller may not setup the B&R mechanism at that time, but may wait for the recovery connection de-activation (meaning a revertive full-rerouting operation) before setting up a B&R mechanism in the dataplane.

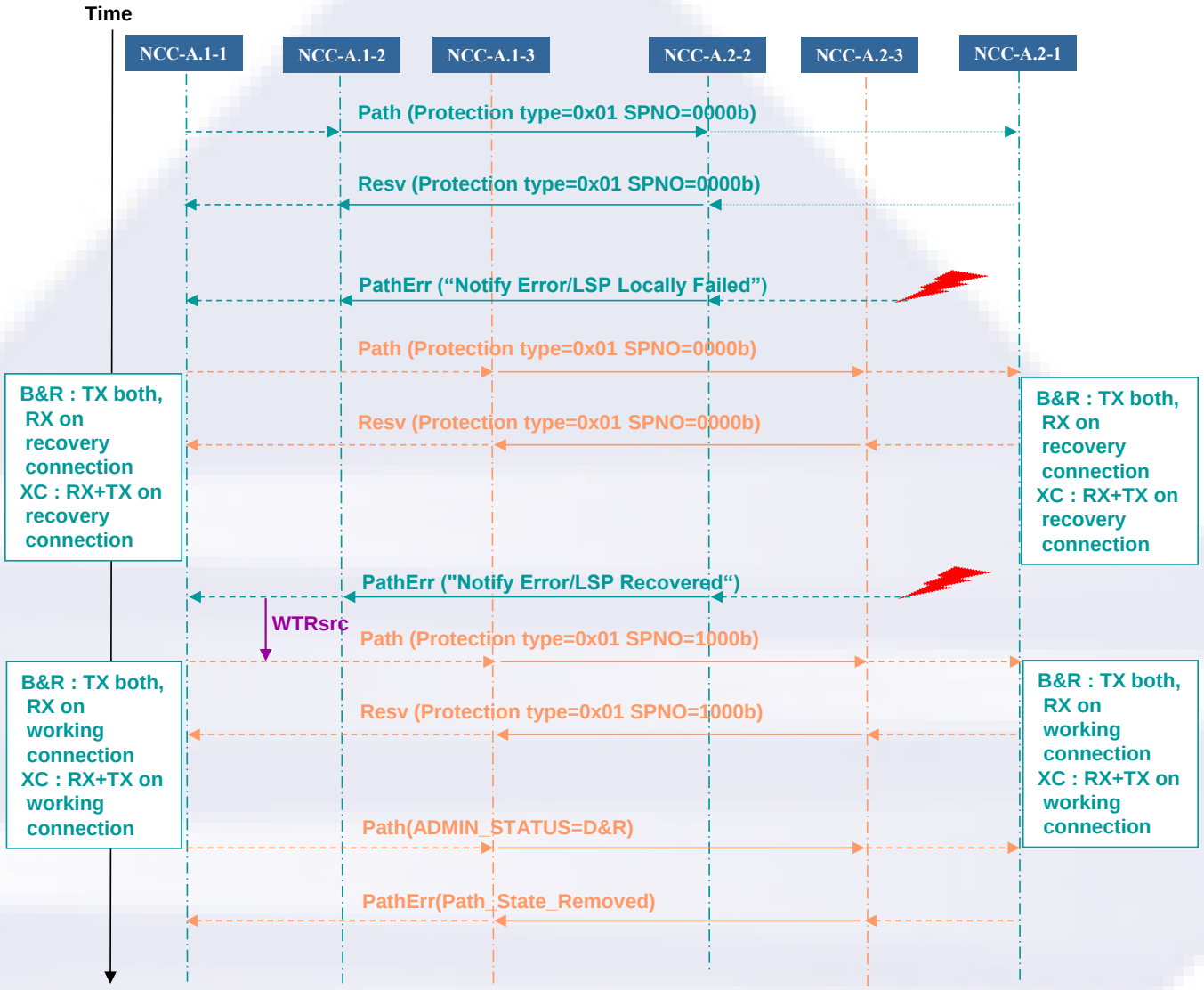


Figure 29: RSVP-TE Signaling for revertive full-rerouting recovery

Figure 29 is based on [OIF-ENNI-REC-AM-01.0] Figure 3 reference network. "BwAR", "B&R" and "XC" are terms defined in [OIF-ENNI-REC-AM-01.0].

6.7.5.4 Non-revertive full-rerouting

In Path messages, the E-NNI Recovery sub-object of an OIF_RECOVERY_STACK object carries a PROTECTION object and an ASSOCIATION object. In Resv messages, it carries a PROTECTION object, no ASSOCIATION object.

As shown in Figure 30, the PROTECTION object is set as follows:

LSP (Protection Type) Flags: 0x01 (Full) Rerouting

Secondary (S) / Protecting (P) / Notification (N) / Operational (O) bits:

- SPNO=0000b for the working connection setup;

- SPNO=0000b for the combined recovery connection setup and activation;

  Note that the SPNO setting for the working and recovery connections setup is no different than the revertive full-rerouting case.

The ASSOCIATION object is set as specified in section 6.3.11.3.2, the Association Type being set to 0x01 (Recovery). A second ASSOCIATION object can be used with the Association Type set to Resource Sharing if resource reuse is desired as specified in section 6.7.3.

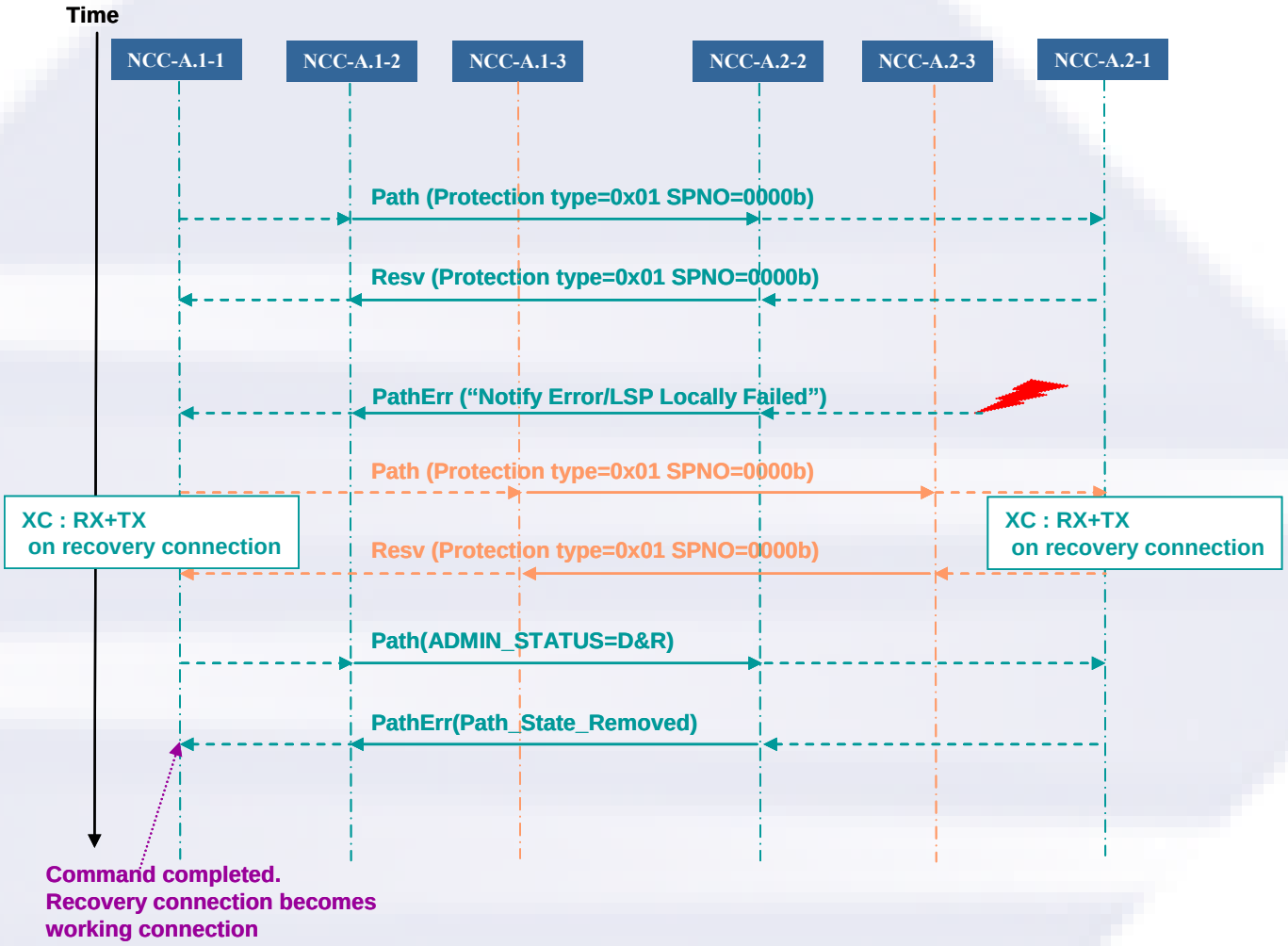


Figure 30: RSVP-TE Signaling for non-revertive full-rerouting recovery

Figure 30 is based on [OIF-ENNI-REC-AM-01.0] Figure 3 reference network. "BwAR", "B&R" and "XC" are terms defined in [OIF-ENNI-REC-AM-01.0].

6.7.6 Reversion

6.7.6.1 Retention of the failed nominal path

When reversion is required, the source node CC must maintain the original connection. This is done by:

- Maintaining the original session (keep sending refresh messages);
- Allowing graceful restart [RFC3473] to re-synchronize the state of the session (if the adjacent CC failed together with the transport plane).

6.7.7 Combination of protection and hard rerouting and soft rerouting

In a recovery domain where a 1+1 protection mechanism has been selected for a call, the working or protecting connection may be hard or soft-rerouted (note: both the working and protecting connections may be soft-rerouted at a given point in time). In such a case, two working connections or two protecting connections will belong to the same recovery association (so there may be up to four connections being signaled in a recovery domain with the same recovery association, at a given point in time)[11]:

- A new working connection (respectively protecting connection) takes precedence over an existing working connection (respectively protecting connection): in case of B&R, the bridge leg must be moved from the existing working connection (respectively protecting connection) to the new working connection (respectively protecting connection).
- Setting the S bit for a working connection (respectively protecting connection) means that this working connection (respectively protecting connection) is de-activated: in case of B&R, the bridge leg must be moved to the other working connection (respectively protecting connection) if it exists.

Moreover, in case of soft-rerouting, the original working or protecting connection MUST be put administratively down by setting the A bit in the ADMIN_STATUS object. In case of temporary soft-rerouting ([OIF-ENNI-REC-AM-01.0]), the ADMIN_STATUS object A bit MUST be reset before or after initiating the reversion process (i.e. de-activating the maintenance connection).

In case of hard-rerouting, the original working or protecting connection SHOULD be put administratively down by setting the A bit in the ADMIN_STATUS object (note that this may not be possible if the failure impacted the control plane too). In case of revertive hard-rerouting, the ADMIN_STATUS object A bit MUST be reset before or after initiating the reversion process (i.e. de-activating the hard-rerouting connection).

Section 11 provides signaling examples for the combination of 1+1 protection and hard/soft-rerouting.

[11] Resource sharing association may also be used but this item is for further study.

7 Compatibility with UNI and E-NNI

Any unknown protocol objects shall be handled according to the methods of their specific protocols.

1) In RSVP-TE, (per [RFC2205]) the class number range has three categories for unknown class
    a. 0-127: the message should be rejected with "Unknown Object Class" error
    b. 128-191: the message should not be rejected but the unknown class should be dropped
    c. 192-255: the message should not be rejected and the unknown class should be forwarded without examination and modification

7.1 Multilayer Amendment Compatibility with UNI

Most of the multilayer amendment extensions [OIF-ENNI-ML-AM-01.0] are transparent to UNI 1.0 and UNI 2.0 implementations. The UNI-Cs SHOULD remain unaware of the multilayer aspects of the network, if applicable but the UNI-C implementations MUST support the new type of source LSR address for the Call ID as described in section 6.3.14. As UNI 1.0 and UNI 2.0 were not explicitly supporting this Call ID format, implementations that do not support this format need to be upgraded to support this Call ID format in order to interwork with a multilayer network.

7.2 Multilayer Amendment Compatibility with E-NNI

The multilayer amendment [OIF-ENNI-ML-AM-01.0] is an extension to E-NNI 2.0. In order to provide multilayer signaling, all nodes at the layer boundaries, i.e. providing adaptation, MUST support this amendment in order to provide multilayer signaling.

Intermediate E-NNI nodes MAY provide only E-NNI 1.0 or E-NNI 2.0 functionality but MUST support the new type of source LSR address for the Call ID as described in section 6.3.14. As E-NNI 1.0 and E-NNI 2.0 were not explicitly supporting this Call ID format, implementations that do not support this format need to be upgraded to support this Call ID format in order to interwork with a multilayer network.

This amendment introduces changes in the Ethernet signaling based on [OIF-UNI-02.0-R2-RSVP] that are not backward compatible with E-NNI 2.0. Implementations of this Implementation Agreement that support Ethernet services MUST support the ability to configure the neighbor's version in order to determine which codepoints and encodings to use for Ethernet based signaling. The version is only used to determine which Ethernet signaling codepoints to use and does not impact other aspects of this amendment.

7.3 Recovery Amendment Compatibility with E-NNI

7.3.1 Compatibility with OIF E-NNI Signaling 1.0

[OIF-E-NNI-sig-01.0] does not support recovery. Indeed, E-NNI 1.0 does not support:

- the NOTIFY_REQUEST object,
- the use of the A-bit in the ADMIN_STATUS object to declare a connection "Administratively DOWN",
- the OIF_RECOVERY_STACK object.

Consequently, there is no backward compatibility between this amendment and [OIF-E-NNI-sig-01.0].

7.3.2 Compatibility with OIF E-NNI Signaling 2.0

An implementation may be compliant to [OIF-E-NNI-sig-02.0], but may not support this recovery amendment.

- To enable recovery for a connection in a domain, the DIN and DEN with regard to that connection must support this amendment;
- To enable resource reuse across a given E-NNI link, both the eNNI-U and eNNI-D nodes must support this amendment;

However, this amendment also requires to be supported by all nodes inside a recovery domain. Indeed, there is an issue with [OIF-E-NNI-sig-02.0]:

- All nodes belonging to a recovery domain need to be able to perform failure notification (see 7.1.6).

This amendment obsoletes the use of the PROTECTION object as described in [OIF-E-NNI-sig-02.0] as it did not include any semantics. The usage of the PROTECTION object is now fully specified as a sub-object of the OIF_RECOVERY_STACK and implementations of this amendment MAY reject requests that include a PROTECTION object not embedded within the OIF_RECOVERY_STACK.

The OIF_VENDOR_PRIVATE_EXTENSION_TYPE_1 vendor private object has been chosen to carry the OIF_RECOVERY_STACK object (section 7.1.1) so that nodes that do not recognize this object will reject a Path message specifying such an object. As a consequence, [OIF-E-NNI-sig-02.0] compliant nodes that do not support this amendment cannot take part in recovery signaling.

The crankback mechanism described below allows some interworking between this amendment and [OIF-E-NNI-sig-02.0].

7.3.3 Crankback

A crankback mechanism may be used to dynamically identify, and go around, nodes that do not support this recovery amendment, when trying to establish working or recovery connections.

[OIF-E-NNI-sig-01.0] or [OIF-E-NNI-sig-02.0] compliant nodes will reject a Path message specifying a OIF_RECOVERY_STACK object with a PathErr message. The PathErr error code should be set to "Unknown object class", and the error value should be set to 0x7f01 (Class number and C-Type of the unknown OIF_RECOVERY_STACK object).

When receiving such a PathErr message, the DIN of the recovery domain may try to compute a new route across its domain, excluding the node that generated the PathErr message.

8 References

Note that in many cases references are self-referential. Instead of "... G.8080 [G.8080]...", the text will state "... [G.8080]..."

8.1 ITU-T

- [G.7042] ITU-T Rec. G.7042 (2006), *Link capacity adjustment scheme (LCAS) for virtual concatenated signals*
- [G.707] ITU-T Rec. G.707 (2003), *Network Node Interface for the Synchronous Digital Hierarchy (SDH)*
- [G.709] ITU-T Rec. G.709 (2003), *Interfaces for the Optical Transport Network (OTN)*
- [G.7712] ITU-T Rec. G.7712/Y.1703 (2003), *Architecture And Specification Of Data Communication Network*
- [G.7713] ITU-T Rec. G.7713/Y.1704 (2001), *Distributed Connection Management (DCM)*, Amendment 1 (2005)
- [G.7713.2] ITU-T Rec. G.7713.2 (2003), *DCM Signalling Mechanism Using GMPLS RSVP-TE (DCM GMPLS RSVP-TE)*
- [G.7715.1] ITU-T Rec. G.7715.1 (2004), *ASON routing architecture and requirements for link state protocols*
- [G.7718] ITU-T Rec. G.7718/Y.1709 (2005), *Framework for ASON management*
- [G.8080] ITU-T Rec. G.8080/Y.1304 (2012), *Architecture of the Automatic Switched Optical Network (ASON)*

8.2 OIF

- [OIF-ENNI2.0-SIG] OIF Implementation Agreement OIF-E-NNI-Sig-02.0 – OIF E-NNI Signaling Specification, April 2009, http://www.oiforum.com/public/documents/OIF_E-NNI_Sig_02.0.pdf
- [OIF-ENNI-ML-AM-01.0] OIF Implementation Agreement, "Multilayer Amendment to E-NNI 2.0 – Common Part", OIF-ENNI-ML-AM-01.0, April 2013, http://www.oiforum.com/public/documents/OIF-ENNI-ML-AM-01.0.pdf
- [OIF-ENNI-REC-AM-01.0] OIF Implementation Agreement, "Recovery Amendment to E-NNI 2.0 – Common Part", OIF-ENNI-REC-AM-01.0,
- [OIF-RSVP-PVT-EXT-01.0] OIF Application of Vendor Private Extensions in RSVP Implementation Agreement, October 2011, http://www.oiforum.com/public/documents/OIF_RSVP_PVT_EXT-01.0.pdf
- [OIF-UNI-02.0] *OIF Implementation Agreement OIF-UNI-02.0-Common - User Network Interface (UNI) 2.0 Signaling Specification: Common Part*, February 2008, http://www.oiforum.com/public/documents/OIF-UNI-02.0-Common.pdf

– [OIF-UNI-02.0-R2-RSVP] *OIF Implementation Agreement OIF-UNI-02.0-R2-RSVP*, "RSVP Extensions for User Network Interface (UNI) 2.0 Signaling", TBD, link TBD

– [OIF-SEC] OIF-SEP-01.0, *Security Extension for UNI and NNI,* May 2003, http://www.oiforum.com/public/documents/Security-IA.pdf.

– [SecAdd] OIF-SEP-02.1 *Addendum to the Security Extension for UNI and NNI*, March 2006, http://www.oiforum.com/public/documents/OIF-SEP-02_1.pdf

8.3 IETF

– [RFC791] IETF RFC 791, *INTERNET PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION*

– [RFC2205] IETF RFC 2205, *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*

– [RFC2747] IETF RFC 2747, *RSVP Cryptographic Authentication*

– [RFC2961] IETF RFC 2961, *RSVP Refresh Overhead Reduction Extensions*

– [RFC3097] IETF RFC 3097, *RSVP Cryptographic Authentication – Updated Message Type Value*

– [RFC3209] IETF RFC 3209, *RSVP-TE: Extensions to RSVP for LSP Tunnels*

– [RFC3471] IETF RFC 3471, *Generalized MPLS - Signaling Functional Description*

– [RFC3473] IETF RFC 3473, *Generalized MPLS Signaling - RSVP-TE Extensions*

– [RFC3474] IETF RFC 3474, *Documentation of IANA Assignments for GMPLS RSVP-TE Usage and Extensions for ASON*

– [RFC3476] IETF RFC 3476, *Documentation of IANA Assignments for LDP, RSVP, and RSVP-TE Extensions for Optical UNI Signaling*

– [RFC3477] IETF RFC 3477, *Signalling Unnumbered Links in RSVP-TE*

– [RFC4328] IETF RFC 4328, *GMPLS Signaling Extensions for G.709 Optical Transport Networks Control*

– [RFC4606] IETF RFC 4606, *GMPLS Extensions for SONET & SDH Control*

– [RFC4872] RSVP-TE Extensions in support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery.

– [RFC4873] GMPLS Segment Recovery

– [RFC4920] IETF RFC 4920, *Crankback Signaling Extensions for MPLS and GMPLS RSVP-TE*

– [RFC4974] IETF RFC 4974, *Generalized MPLS (GMPLS) RSVP-TE Signaling Extensions*

– [RFC5612] IETF RFC 5612, *Enterprise Number for Documentation Use*

– [RFC6003] IETF RFC 6003, *Ethernet Traffic Parameters*

– [RFC6689] IETF RFC 6689 *Usage of the RSVP Association Object*

– [RFC6780] IETF RFC 6780 *RSVP ASSOCIATION Object Extensions*

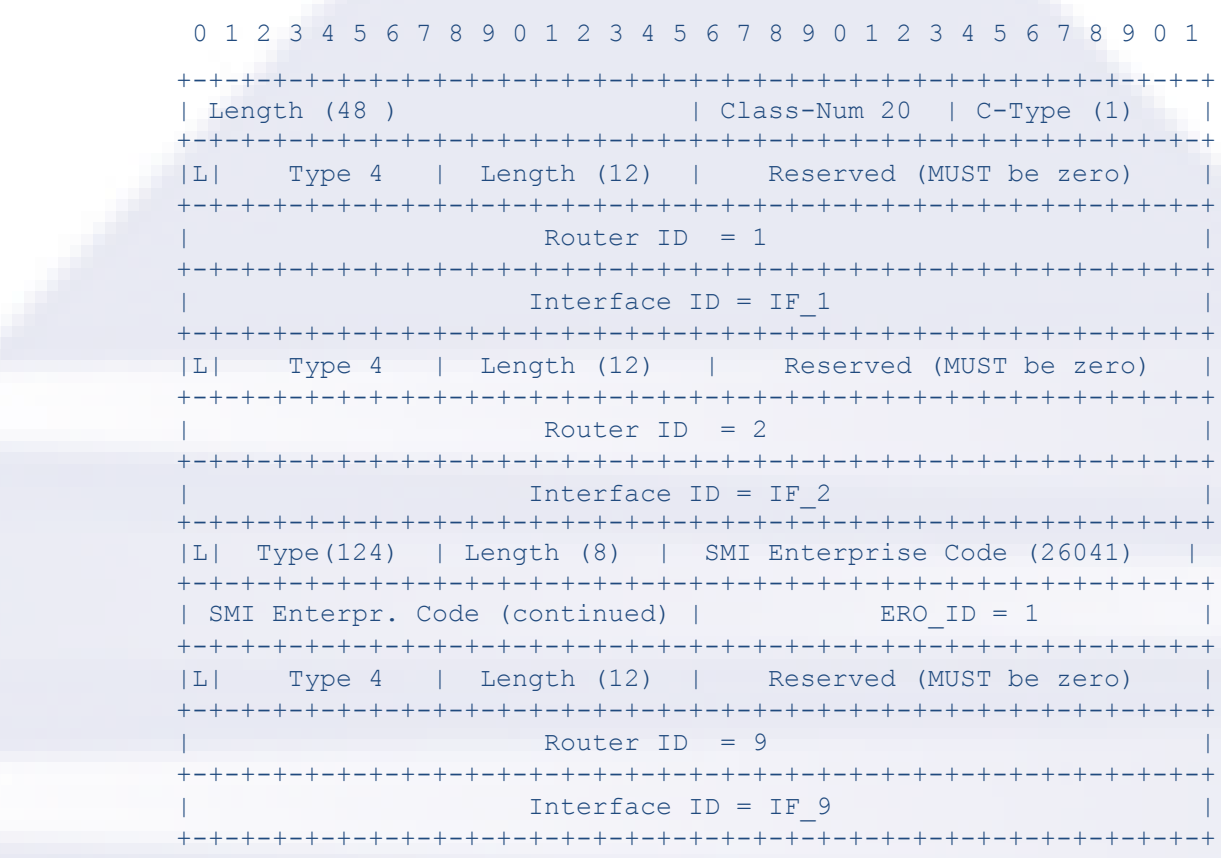
8.4 T1X1.5

– [T1.105] ANSI T1.105 (1995), *Synchronous Optical Network (SONET) – Basic Description including Multiplex Structure, Rates and Formats*

9 Appendix I: Example Nested ERO/RRO

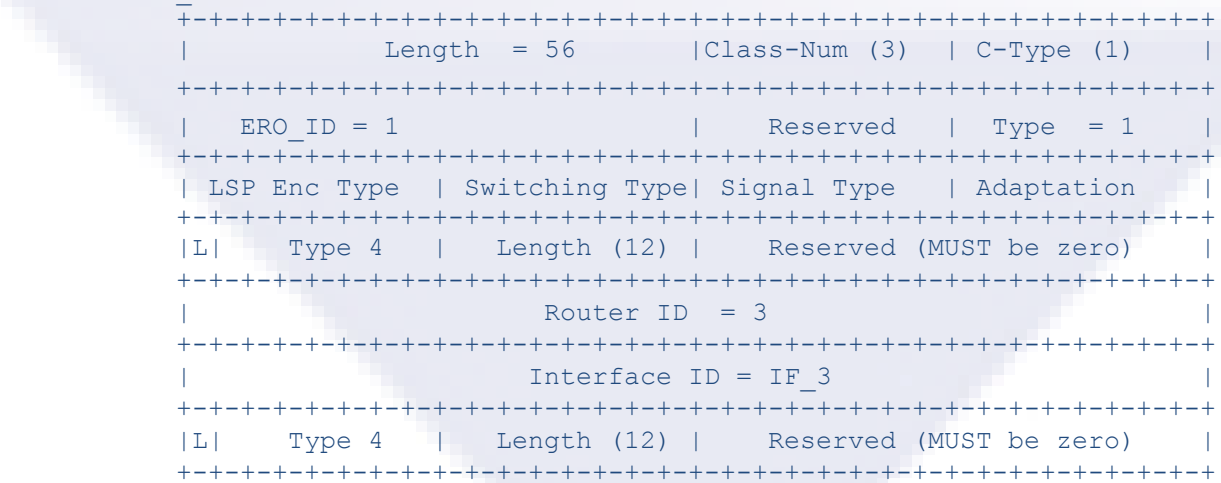
9.1 Example Nested ERO

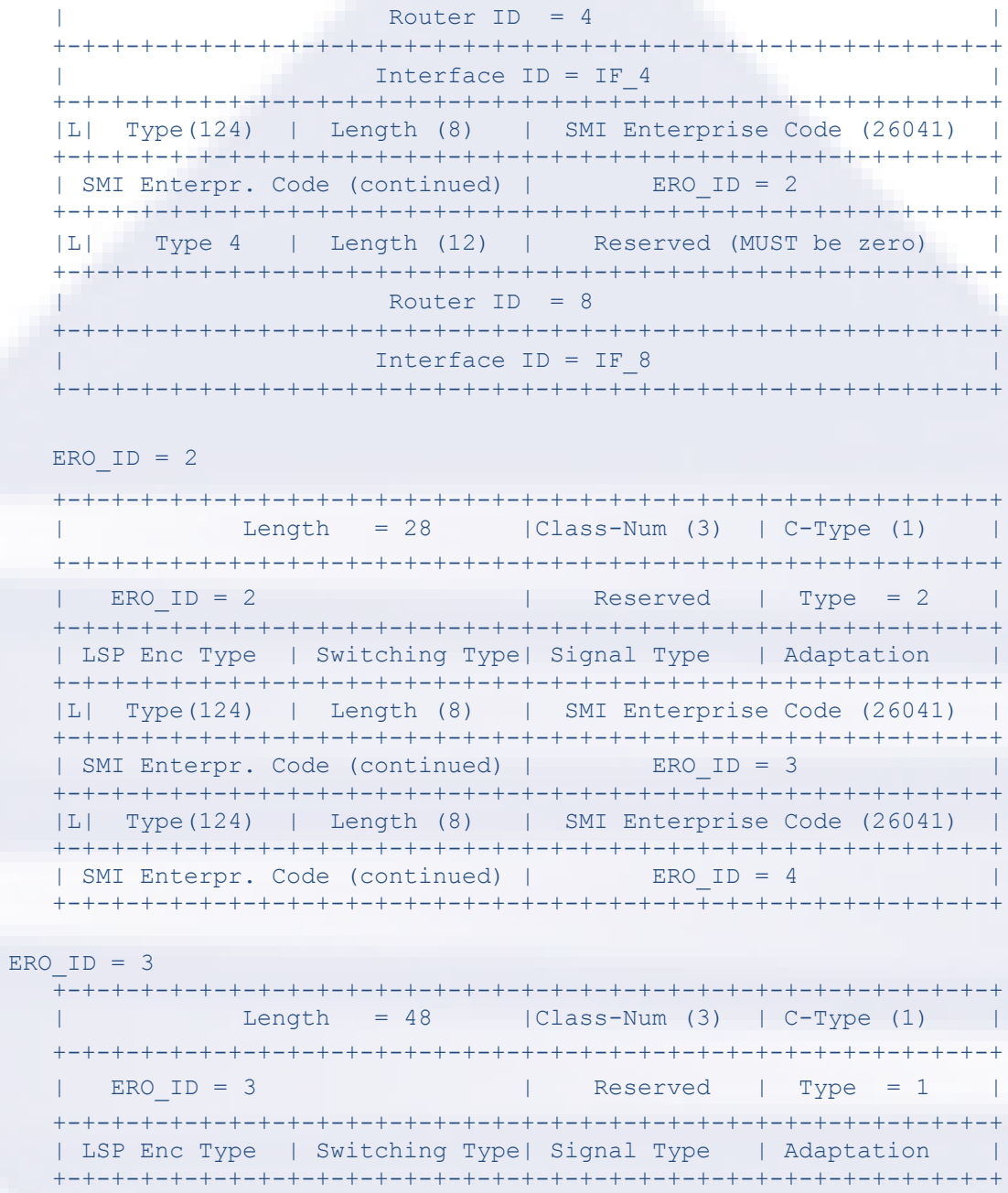
The following example represents an ERO based on Figure 2.

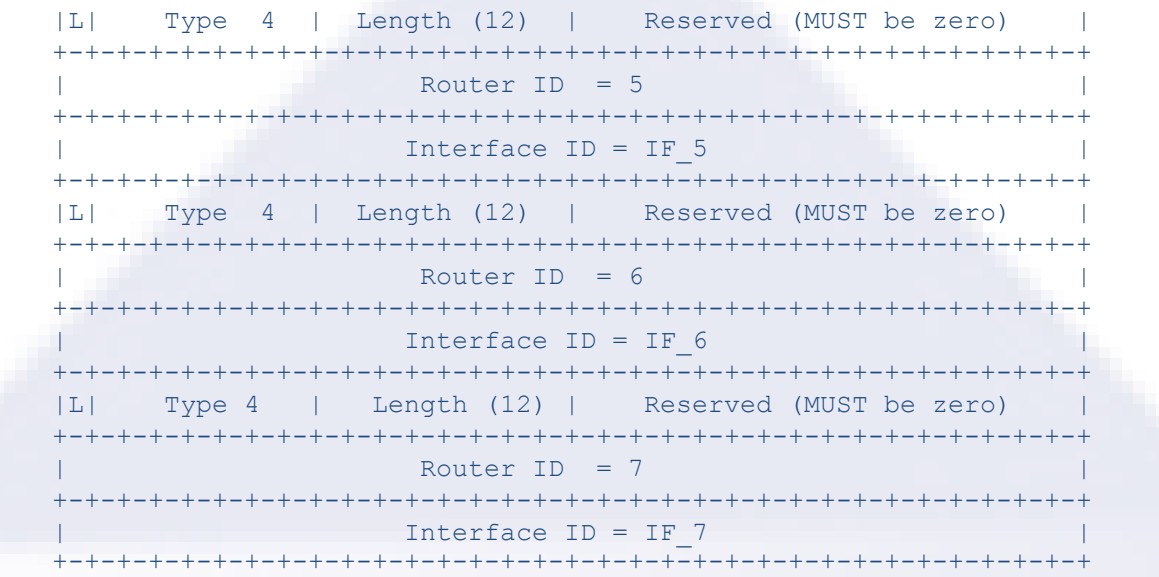


The following sub-objects are included in the
OIF_VENDOR_PRIVATE_EXTENSION_TYPE_3

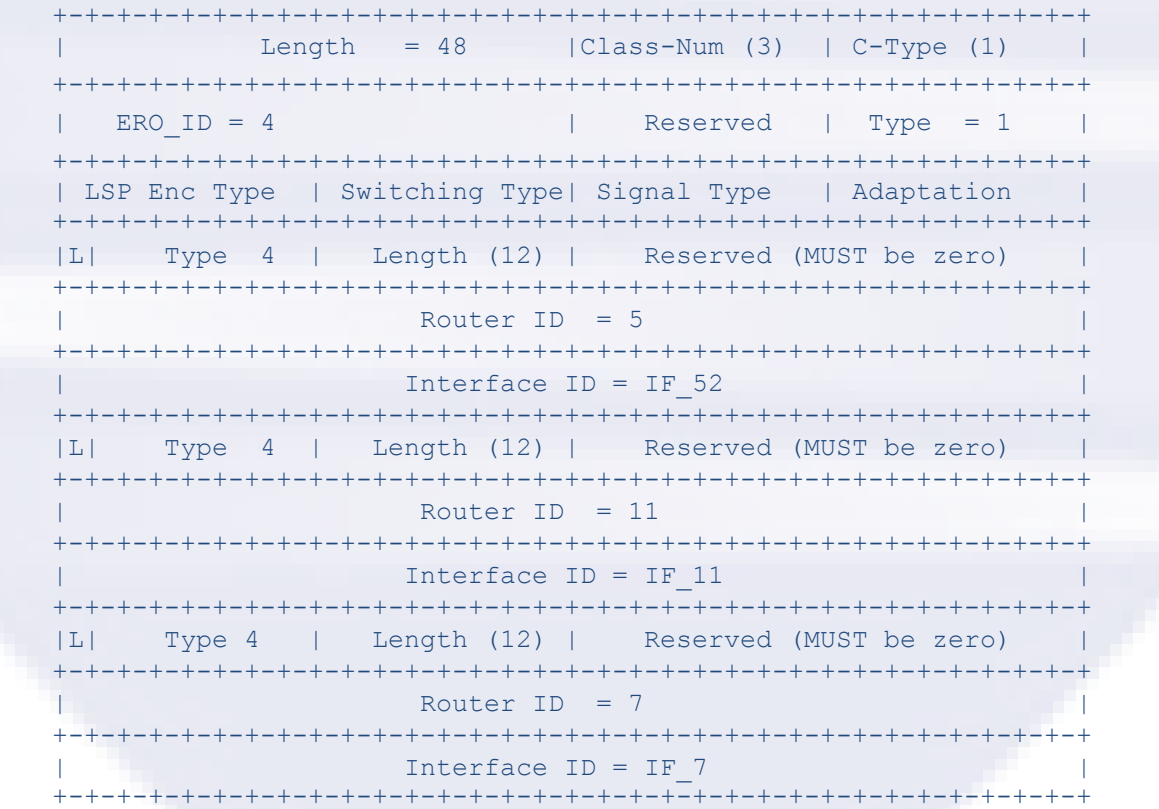
ERO_ID = 1




Router ID = 4
Interface ID = IF_4
L | Type(124) | Length (8) | SMI Enterprise Code (26041)
SMI Enterpr. Code (continued) | ERO_ID = 2
L | Type 4 | Length (12) | Reserved (MUST be zero)
Router ID = 8
Interface ID = IF_8
ERO_ID = 2
Length = 28
Class-Num (3)
C-Type (1)
ERO_ID = 2
Reserved
Type = 2
LSP Enc Type
Switching Type
Signal Type
Adaptation
L | Type(124) | Length (8) | SMI Enterprise Code (26041)
SMI Enterpr. Code (continued) | ERO_ID = 3
L | Type(124) | Length (8) | SMI Enterprise Code (26041)
SMI Enterpr. Code (continued) | ERO_ID = 4
ERO_ID = 3
Length = 48
Class-Num (3)
C-Type (1)
ERO_ID = 3
Reserved
Type = 1
LSP Enc Type
Switching Type
Signal Type
Adaptation

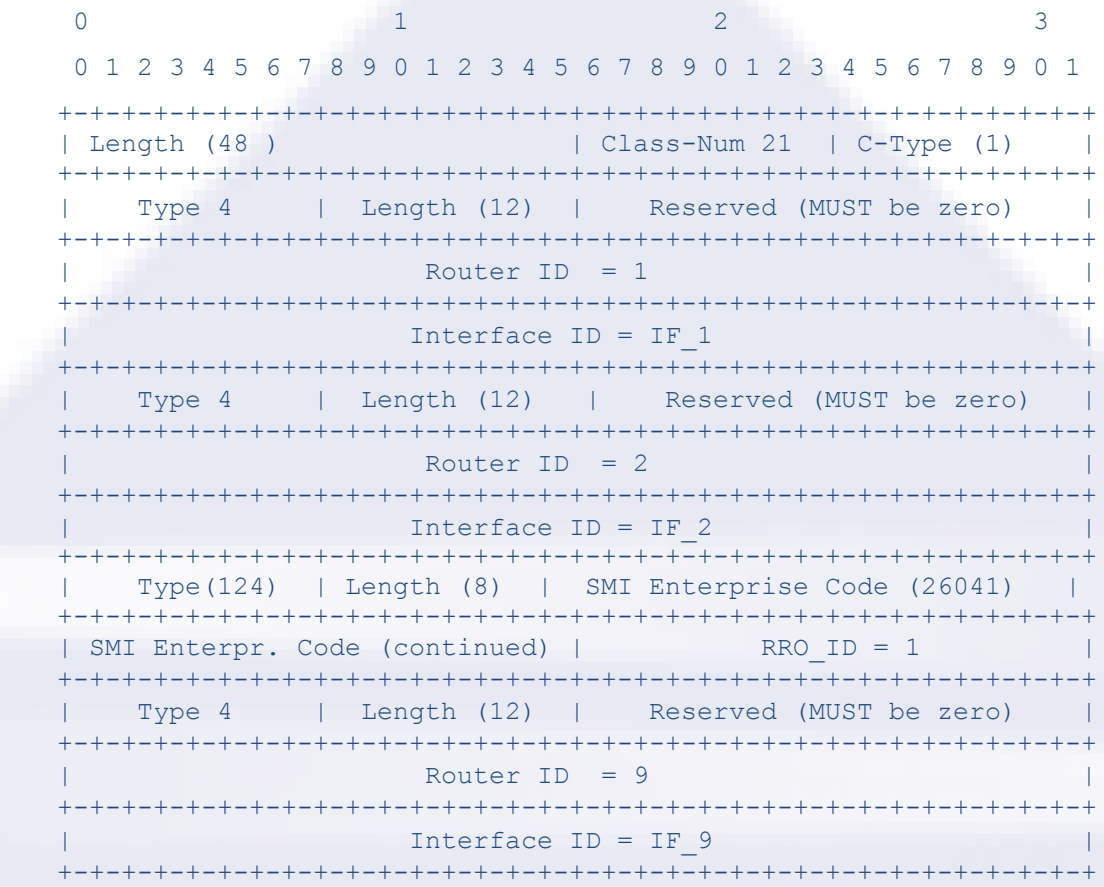


ERO_ID = 4



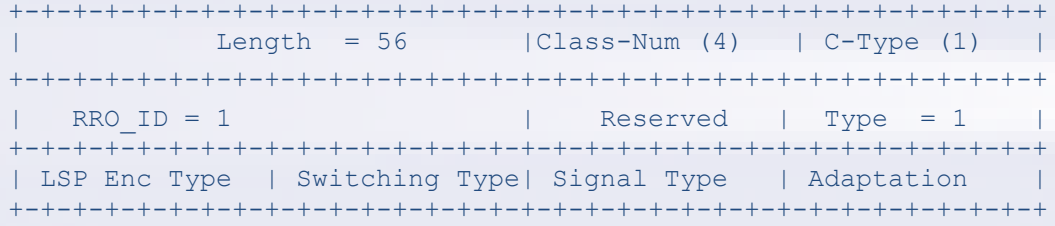
9.2 Example Nested RRO

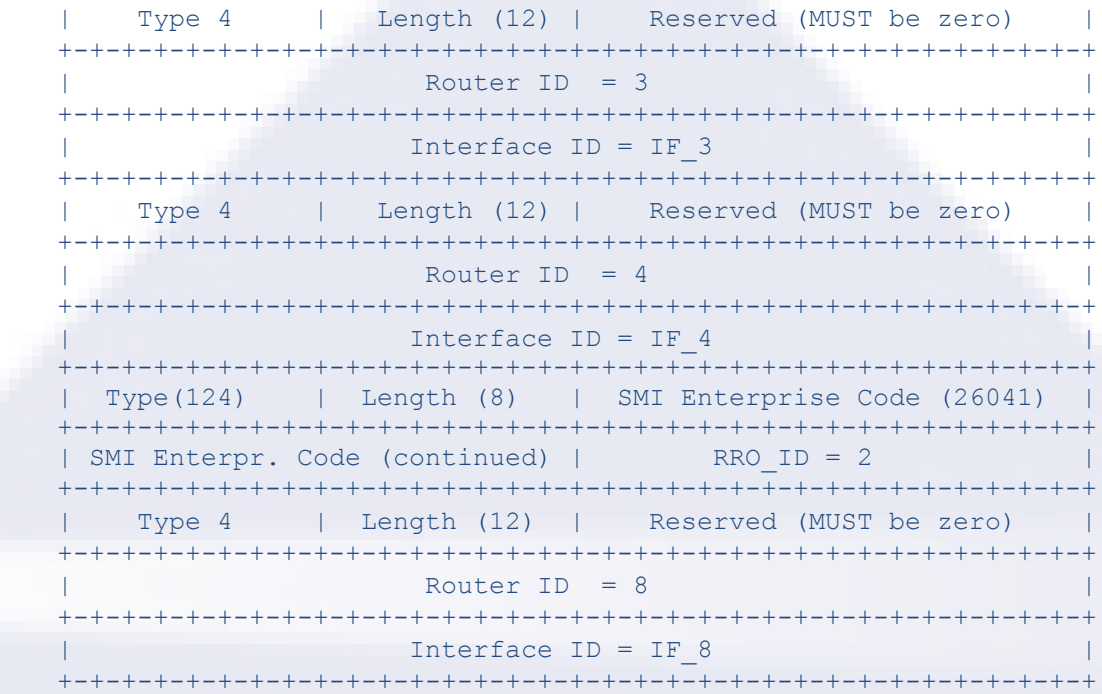
The following example represents an RRO based on Figure 2.



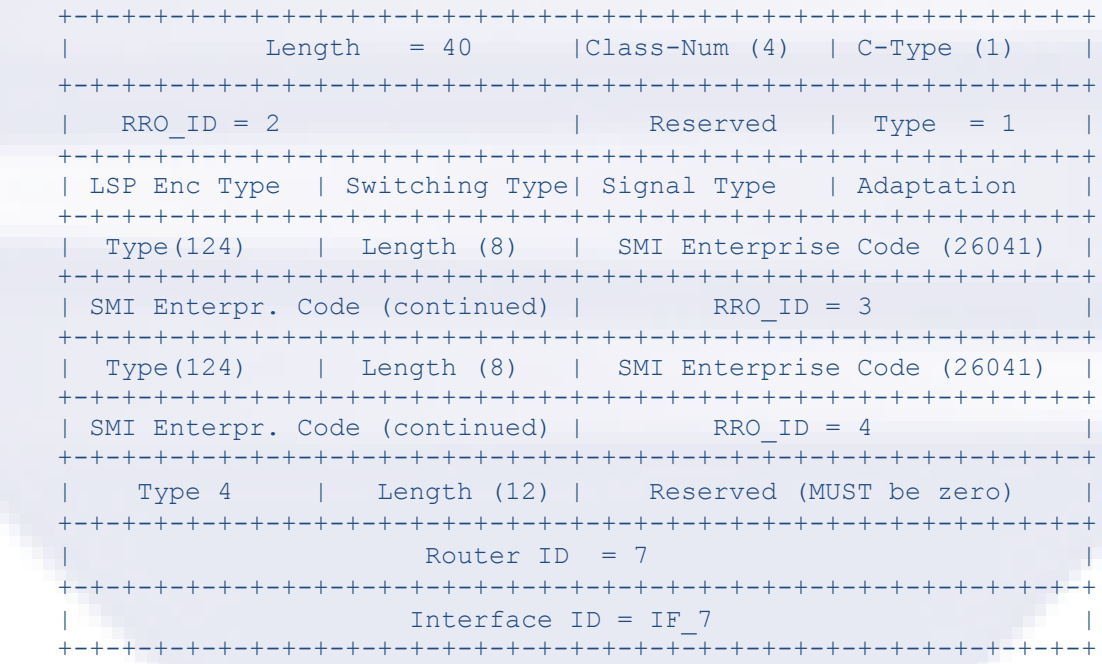
The following sub-objects are included in the
OIF_VENDOR_PRIVATE_EXTENSION_TYPE_3

RRO_ID = 1

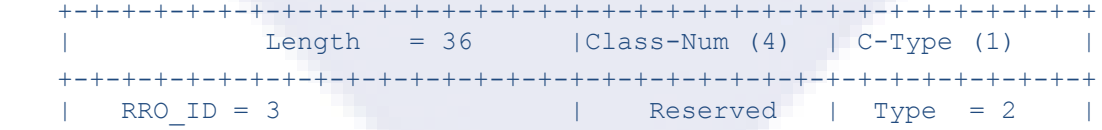


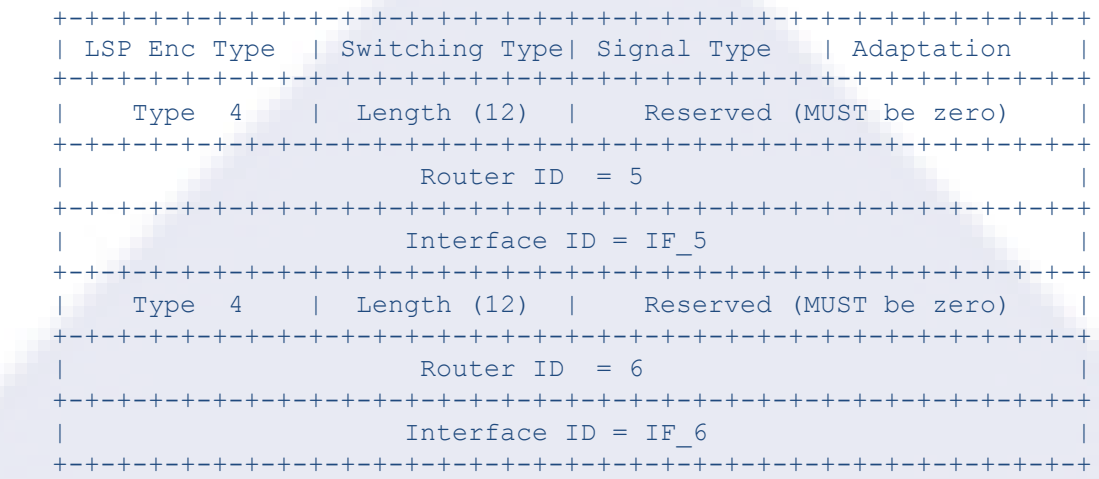


RRO_ID = 2

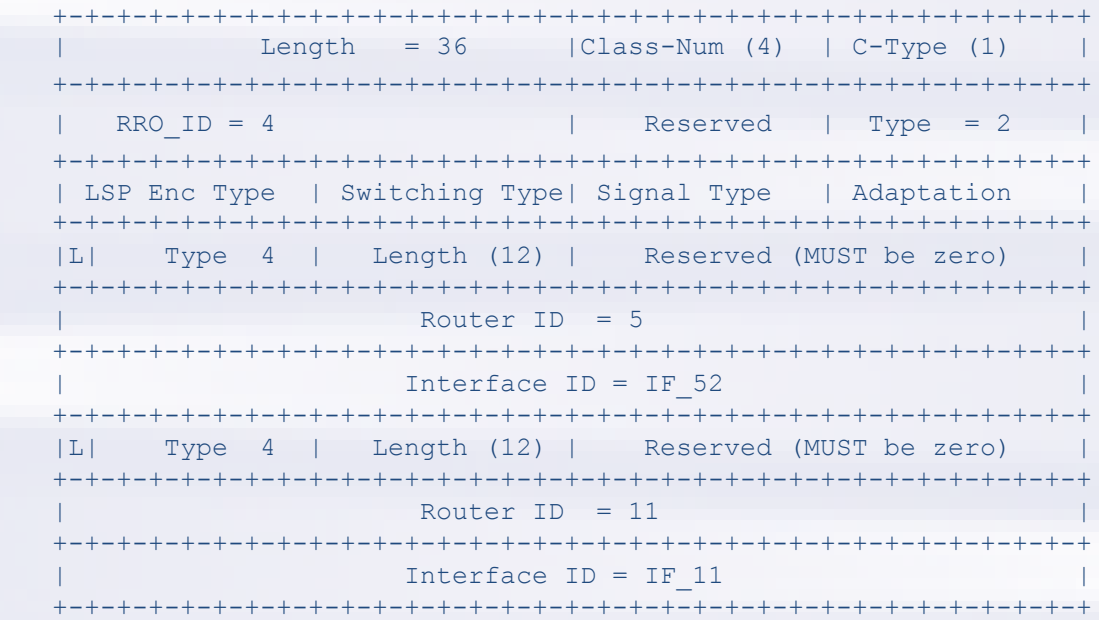


RRO_ID = 3




| LSP Enc Type | Switching Type| Signal Type | Adaptation |
| Type 4 | Length (12) | Reserved (MUST be zero) |
Router ID = 5
Interface ID = IF_5
| Type 4 | Length (12) | Reserved (MUST be zero) |
Router ID = 6
Interface ID = IF_6

RRO_ID = 4


Length = 36
Class-Num (4)
C-Type (1)
RRO_ID = 4
Reserved
Type = 2
| LSP Enc Type | Switching Type| Signal Type | Adaptation |
|L| Type 4 | Length (12) | Reserved (MUST be zero) |
Router ID = 5
Interface ID = IF_52
|L| Type 4 | Length (12) | Reserved (MUST be zero) |
Router ID = 11
Interface ID = IF_11

10 Appendix II: Summary of Multilayer Extensions

The following extensions were made as part of the Multilayer Amendment:

- Added OIF_VENDOR_PRIVATE_EXTENSION_TYPE_1 and OIF_VENDOR_PRIVATE_EXTENSION_TYPE_3 to the Path message in section 6.2.2, to the Resv message in section 6.2.3 and defined new sub-objects in Table 5, Table 6, sections 6.3.11 and 6.3.12
- Added support for the INTSERV_TPSEC and INTSERV_FLOWSPEC in sections 6.2.2, 6.2.3, 6.3.10, Table 5 and Table 6.
- Added support for inverse multiplexing and transitional link support in the ERO and RRO in sections 6.3.2, 6.3.3 and Table 6.
- Clarified usage of SPC_LABEL for multilayer in section 6.3.4.
- Added support for the IPv4 RSVP_HOP in section 6.3.6.
- Added support for VCAT labels in section 6.3.13.
- Defined a new Source LSR address type for the Call ID in section 6.3.14.
- Defined Layer Identifier in section 6.3.15 and Adaptation in section 6.3.16.
- Added new Call Modification details for VCAT in sections 6.4.2 and subsections 6.4.2.2 and 6.4.2.3.
- Addressed Multilayer compatibility with UNI and E-NNI in sections 7.1 and 7.2.
- Updated references to OIF documents and added a reference to IETF RFC6003 in section 8.
- Added an example nested ERO/RRO encoding in section 9.

11 Appendix III:: Combined 1+1 protection and soft/hard rerouting

This appendix provides signaling examples for combined 1+1 protection and soft/hard rerouting. This appendix does not provide an exhaustive list of all possible ways for the DIN of a recovery domain to combine 1+1 protection and soft/hard rerouting.

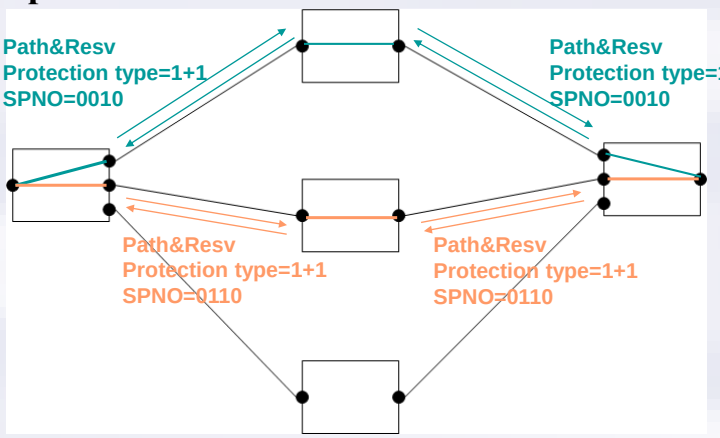
11.1 Combined 1+1 protection and soft rerouting

Figure 31 shows a signaling example to perform a soft-rerouting of a working connection.

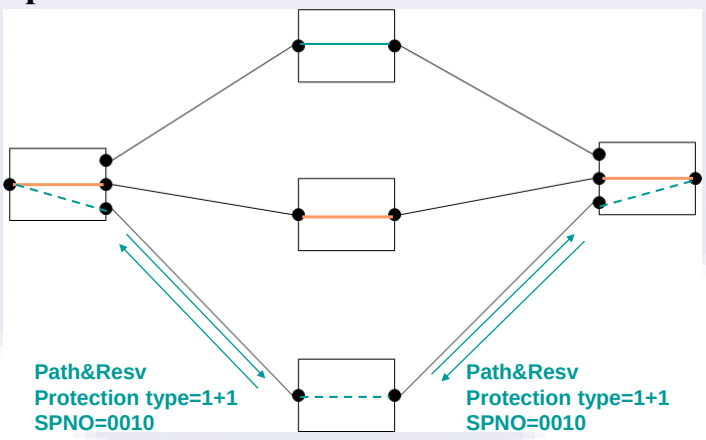
- Step 1: the 1+1 protection working and protecting connections are established as usual;
- Step 2: a third connection is established for maintenance purpose. It can be a working or a protecting connection depending on which one is being soft-rerouted (the working one has been chosen on Figure 31).
- Step 3: the ADMIN_STATUS A bit is set for the original (soft-rerouted) working or protecting connection.

Note: Step 3 may be performed before Step 2. Or Step 3 and Step 2 may be performed concurrently. Receiving implementation should handle any ordering.

Step 1



Step 2



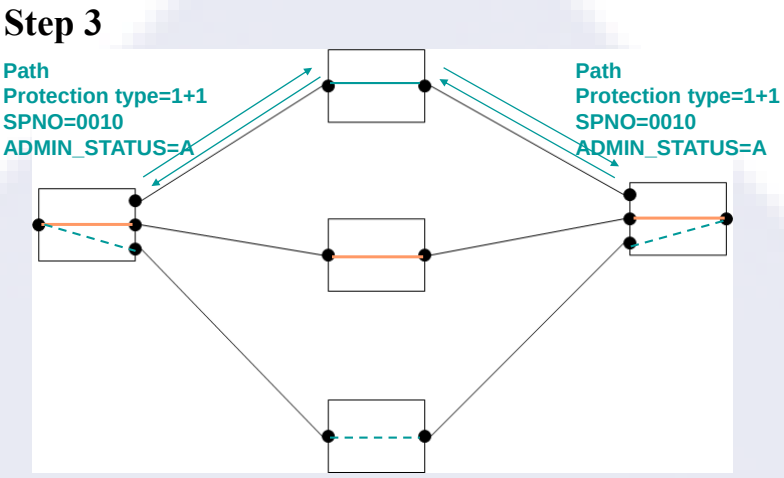


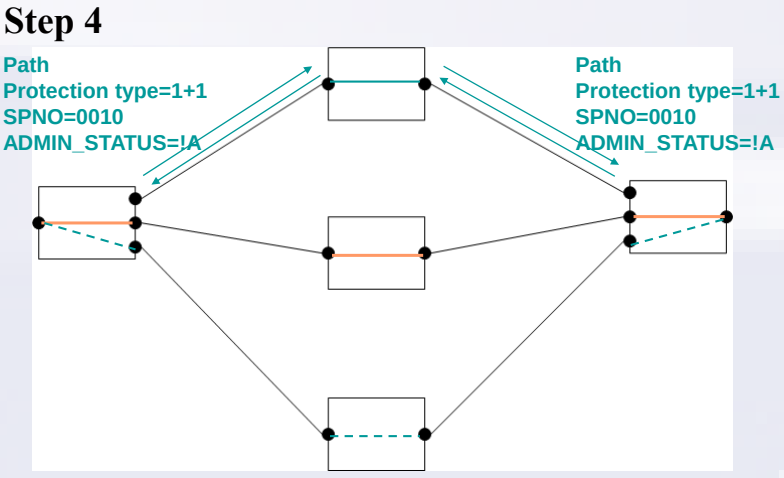
Figure 31: Combined 1+1 protection and soft-rerouting

At this stage, if this is a permanent soft-reroute operation, the original working or protecting connection is torn down. Otherwise, the connection is maintained for reversion at a later stage.

For reversion (Figure 32):

- Step 4: the ADMIN_STATUS A bit is unset for the original (soft-rerouted) working or protecting connection.
- Step 5:the maintenance connection is de-activated.
- Then the maintenance connection can be torn down.

Note: Step 5 may be performed before Step 4.



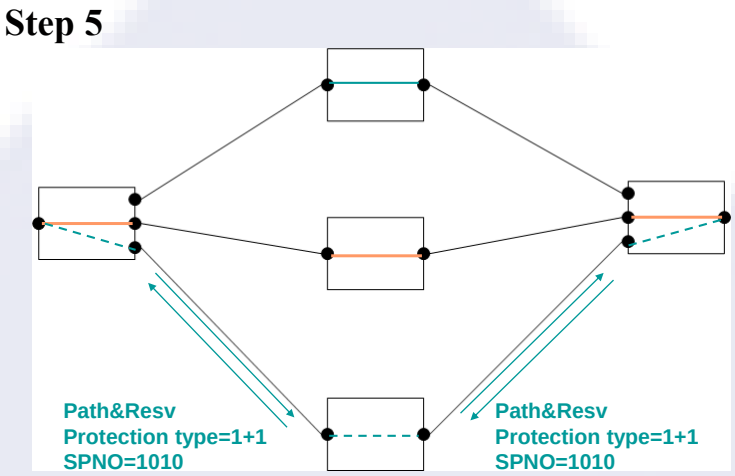
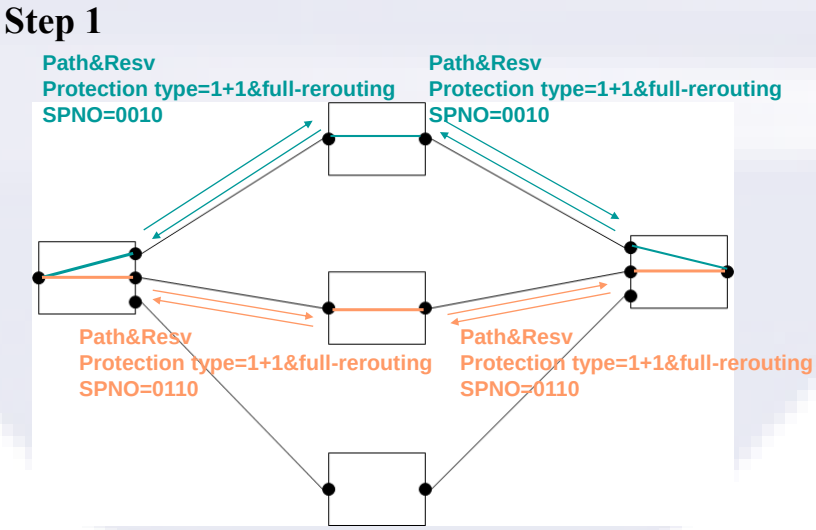


Figure 32: Reversion for combined 1+1 protection and soft-rerouting

11.2 Combined 1+1 protection and hard rerouting: "always on"

Figure 33 shows an example of a signaling sequence after the failure of the working connection:

- Step 1: The 1+1 protection working and recovery connections are established; note that the LSP (Protection Type) Flags field (PROTECTION object) is set either to 0x09 or 0x11.
- Step 2: When a failure occurs for one of the working or protecting connection, it is notified to the recovery domain DIN. Note that if the failure occurs for the working connection, the recovery one will be activated as described in section 6.7.5.1, but this is not shown on Figure 33 (this figure chooses to show a working connection failure).
- Step 3: Hard-rerouting may be triggered (depending on the DIN local policies).



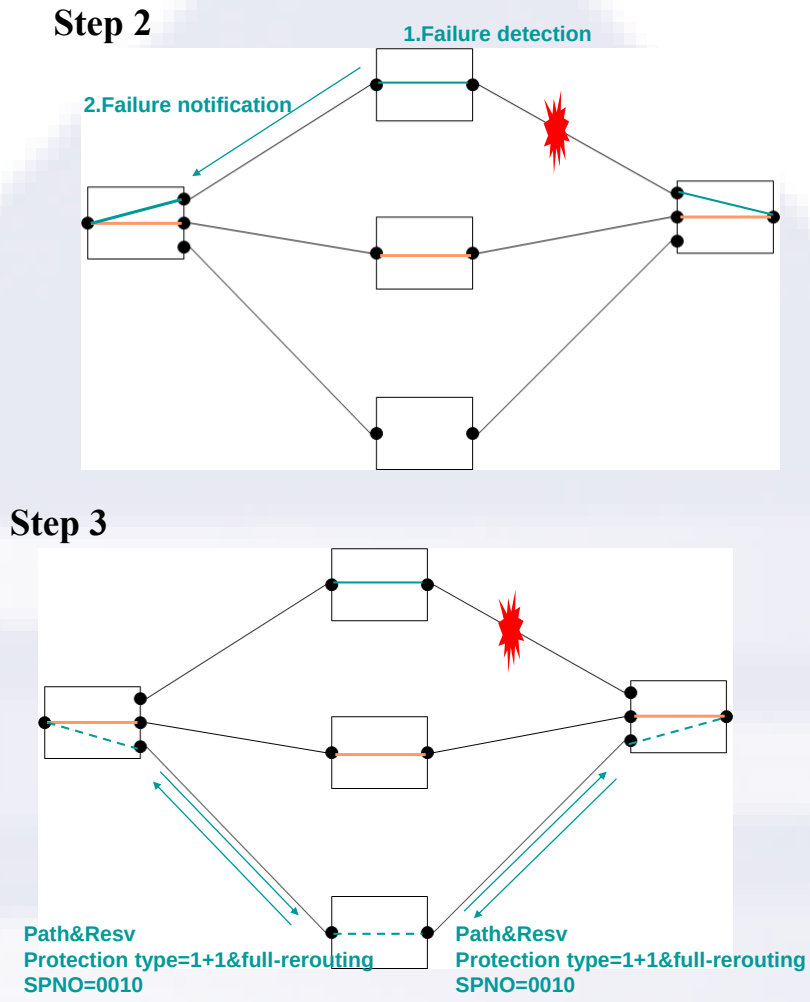


Figure 33: Combined protection and hard-rerouting ("always on protection")

For reversion, the hard-rerouting connection may be first de-activated (similar to Figure 32). Then the hard-rerouting connection can be torn down.

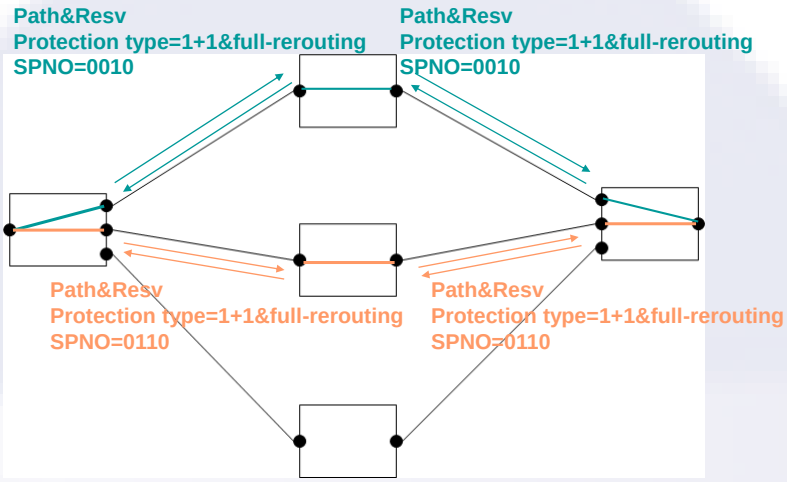
11.3 Combined 1+1 protection and hard rerouting: "2nd level restoration"

Figure 34 shows an example of a signaling sequence where the DIN chooses to hard-reroute the protecting connection after both the working and protecting connections have failed:

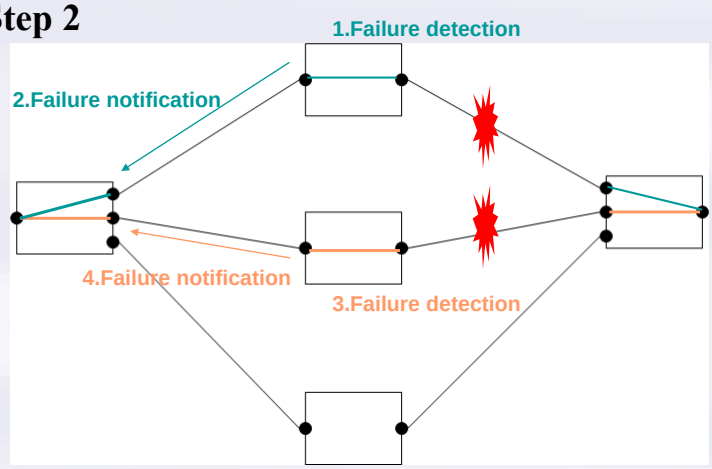
- Step 1: The 1+1 protection working and recovery connections are established; note that the LSP (Protection Type) Flags field (PROTECTION object) is set either to 0x09 or 0x11;
- Step 2: When a first failure occurs (impacting either the working or protecting connection), no hard-rerouting is triggered. Note that if the failure occurs for the working connection, the recovery one will be activated as described in section 6.7.5.1, but this is not shown on Figure 34. Only when a second failure occurs (impacting the other working or protecting connection), hard-rerouting is triggered (Step 3). It is a local decision at the

DIN to restore either the working or protecting connection. Figure 34 chooses to show the restoration of the protecting connection

Step 1



Step 2



Step 3

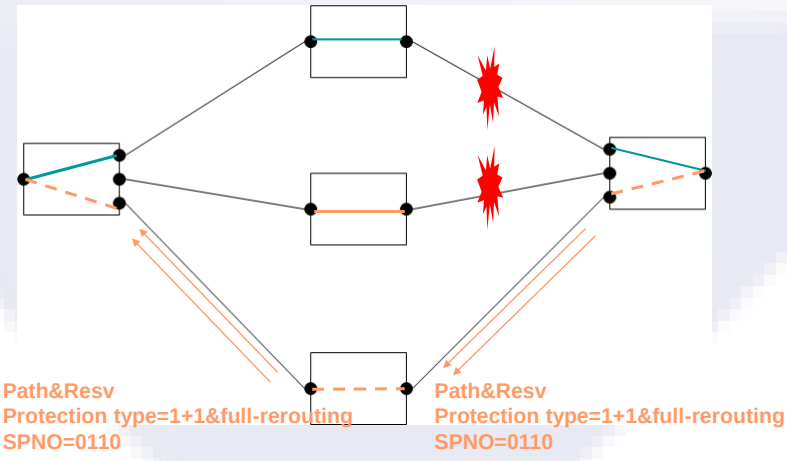


Figure 34: Combined protection and hard-rerouting (“2nd level restoration”)

For reversion, the hard-rerouting connection may be first de-activated (Figure 35). Then the hard-rerouting connection can be torn down.

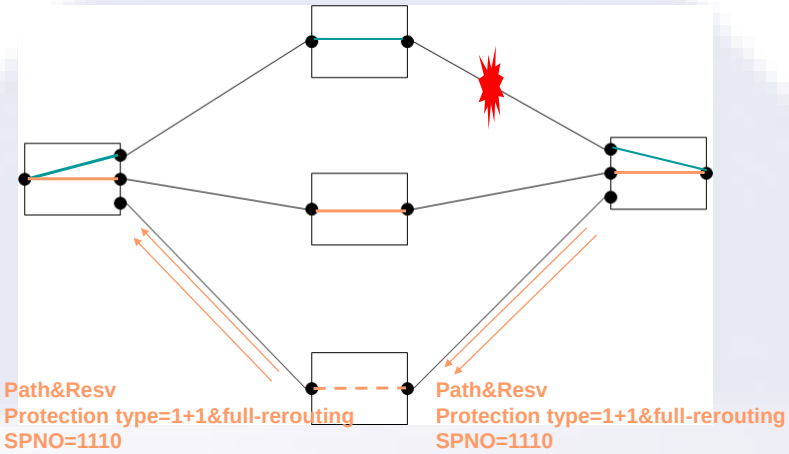


Figure 35: Combined protection and hard-rerouting (“2nd level restoration”). Reversion.

12 Appendix IV:: Examples OIF_RECOVERY_STACK object

This appendix provides OIF_RECOVERY_STACK object encoding examples. They are based on **Figure 36**.

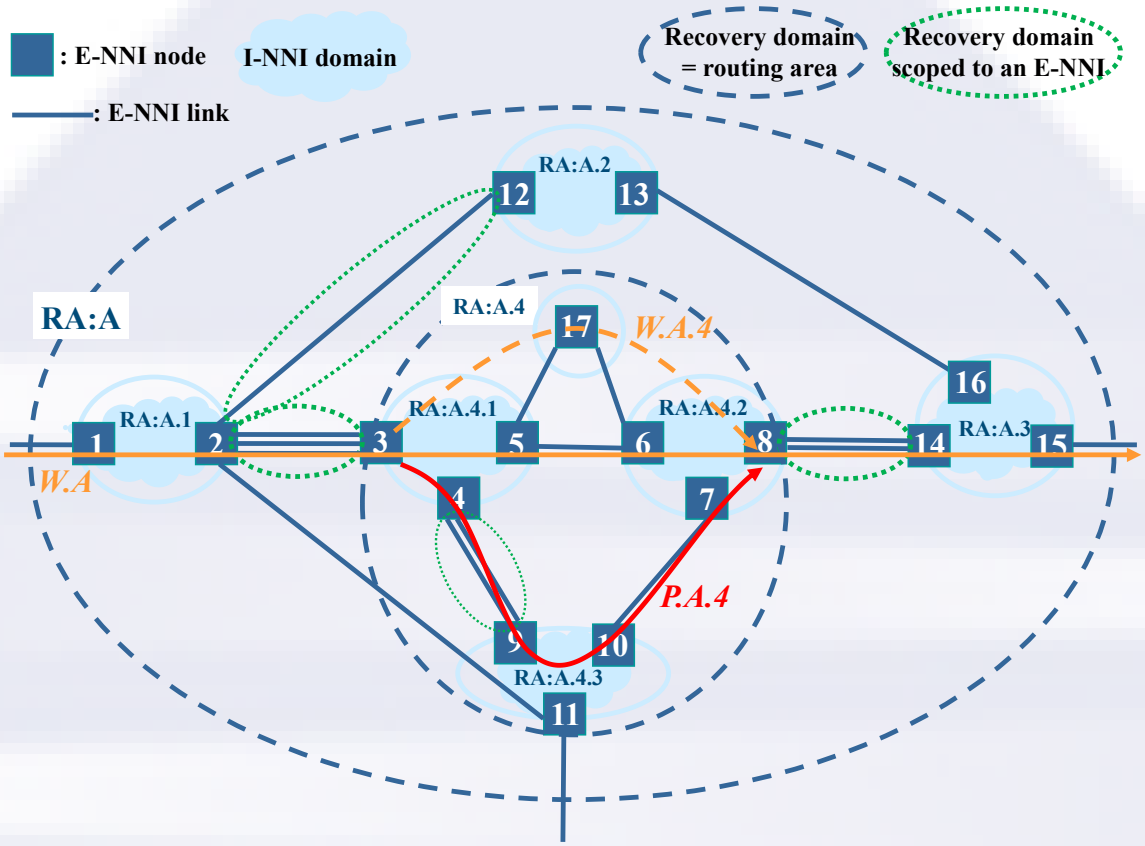


Figure 36: OIF_RECOVERY_STACK examples

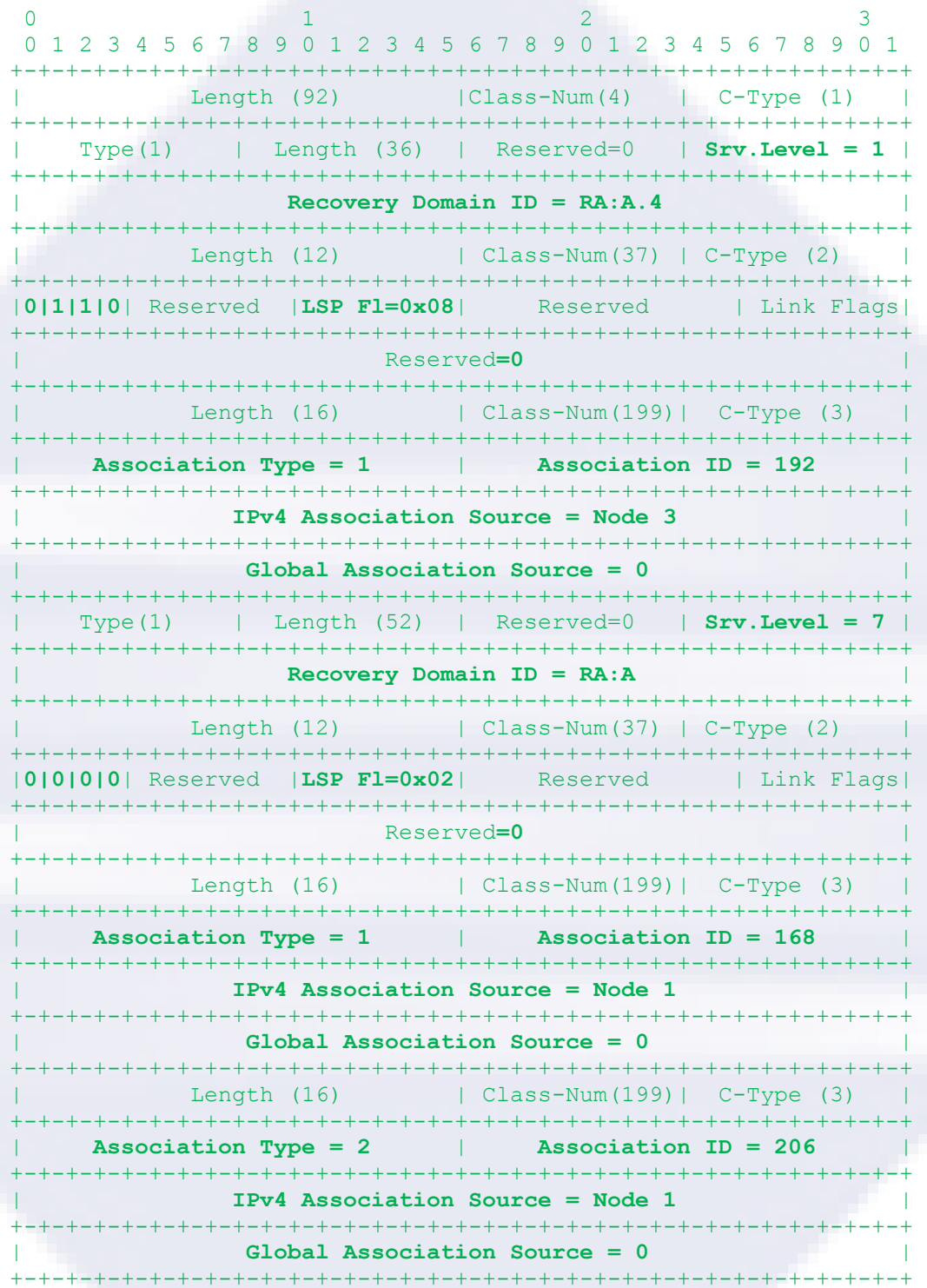
12.1 Within a nested recovery domain

The following OIF_RECOVERY_STACK object encoding assumes:

- There was no OIF_RECOVERY_STACK object specified by the Path message received by node 1 for connection ***W.A***;
- That node 1 (DIN for recovery domain RA:A), based on its local policies and the G.UNI service level, selected:
  - "Revertive hard-rerouting" for the recovery mechanism within recovery domain RA:A; Resource re-use across E-NNI links and within I-NNI domains is required;
  - 7 as recovery domain RA:A "local domain service level";

- That node 3 (DIN for recovery domain RA:A.4), based on its local policies and the "local domain service level" specified by the top E-NNI Recovery sub-object (i.e. 7) in the received OIF_RECOVERY_STACK object, selected:
    - "1+1 Protection" for the recovery mechanism within recovery domain RA:A.4;
    - 1 as recovery domain RA:A.4 "local domain service level";

Then the encoding of the OIF_RECOVERY_STACK specified in the Path message sent across recovery domain RA:A.4 for the protecting connection ***P.A.4*** is as follows:

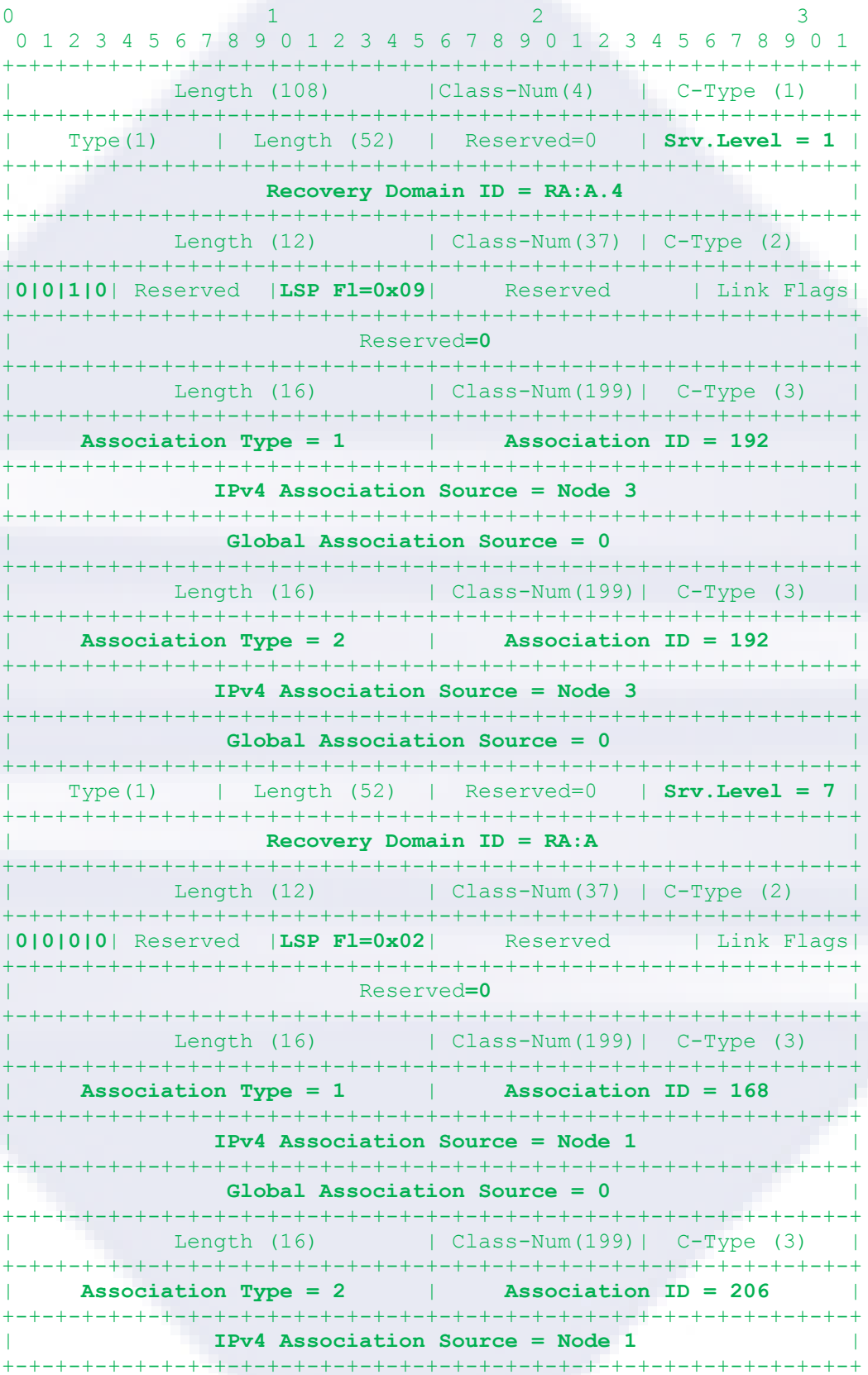

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Length (92) | Class-Num(4) | C-Type (1)
Type(1) | Length (36) | Reserved=0 | Srv.Level = 1
Recovery Domain ID = RA:A.4
Length (12) | Class-Num(37) | C-Type (2)
0|1|1|0| Reserved |LSP Fl=0x08| Reserved | Link Flags|
Reserved=0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 1 | Association ID = 192
IPv4 Association Source = Node 3
Global Association Source = 0
Type(1) | Length (52) | Reserved=0 | Srv.Level = 7
Recovery Domain ID = RA:A
Length (12) | Class-Num(37) | C-Type (2)
0|0|0|0| Reserved |LSP Fl=0x02| Reserved | Link Flags|
Reserved=0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 1 | Association ID = 168
IPv4 Association Source = Node 1
Global Association Source = 0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 2 | Association ID = 206
IPv4 Association Source = Node 1
Global Association Source = 0

12.2 Protection and soft-rerouting combination

The operator of recovery domain RA:A.4 later chooses to trigger a soft-rerouting of the working leg (through the abstract node 17, the soft-rerouting connection being shown as W.A.4 in **Figure 36**).

Resource re-use across E-NNI links and within I-NNI domains is required.

The encoding of the OIF_RECOVERY_STACK specified in the Path message sent across recovery domain RA:A.4 for the soft-rerouting/working connection ***W.A.4*** is as follows:

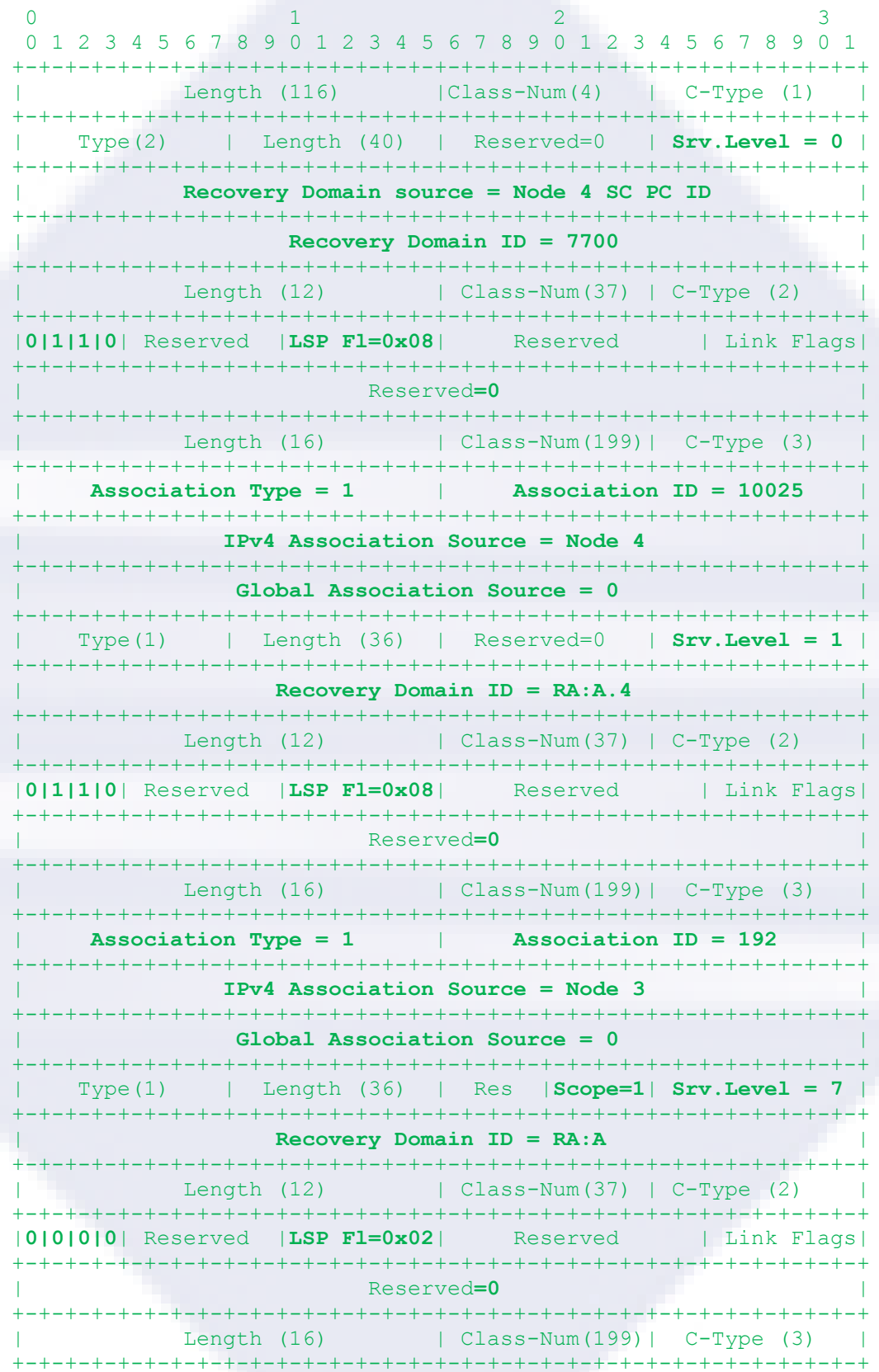

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Length (108) | Class-Num(4) | C-Type (1)
Type(1) | Length (52) | Reserved=0 | Srv.Level = 1
Recovery Domain ID = RA:A.4
Length (12) | Class-Num(37) | C-Type (2)
0|0|1|0| Reserved | LSP Fl=0x09 | Reserved | Link Flags
Reserved=0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 1 | Association ID = 192
IPv4 Association Source = Node 3
Global Association Source = 0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 2 | Association ID = 192
IPv4 Association Source = Node 3
Global Association Source = 0
Type(1) | Length (52) | Reserved=0 | Srv.Level = 7
Recovery Domain ID = RA:A
Length (12) | Class-Num(37) | C-Type (2)
0|0|0|0| Reserved | LSP Fl=0x02 | Reserved | Link Flags
Reserved=0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 1 | Association ID = 168
IPv4 Association Source = Node 1
Global Association Source = 0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 2 | Association ID = 206
IPv4 Association Source = Node 1

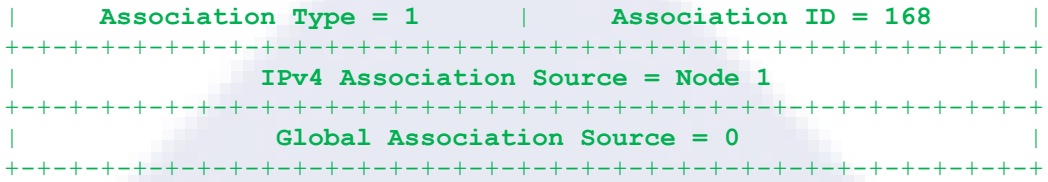
```
|                Global Association Source = 0                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

12.3 Recovery across an E-NNI scoped recovery domain

The recovery connection within recovery domain RA:A.4 may itself be protected when it crosses the E-NNI scoped recovery domain between Node 4 and Node 9. Assuming no 1+1 APS link, two working and protecting connections will be signaled between Node 4 (DIN) and Node 9 (DEN).

The encoding of the OIF_RECOVERY_STACK specified in the Path message for the protecting connection is as follows:


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Length (116) | Class-Num(4) | C-Type (1)
Type(2) | Length (40) | Reserved=0 | Srv.Level = 0
Recovery Domain source = Node 4 SC PC ID
Recovery Domain ID = 7700
Length (12) | Class-Num(37) | C-Type (2)
0|1|1|0| Reserved |LSP Fl=0x08| Reserved | Link Flags
Reserved=0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 1 | Association ID = 10025
IPv4 Association Source = Node 4
Global Association Source = 0
Type(1) | Length (36) | Reserved=0 | Srv.Level = 1
Recovery Domain ID = RA:A.4
Length (12) | Class-Num(37) | C-Type (2)
0|1|1|0| Reserved |LSP Fl=0x08| Reserved | Link Flags
Reserved=0
Length (16) | Class-Num(199) | C-Type (3)
Association Type = 1 | Association ID = 192
IPv4 Association Source = Node 3
Global Association Source = 0
Type(1) | Length (36) | Res |Scope=1| Srv.Level = 7
Recovery Domain ID = RA:A
Length (12) | Class-Num(37) | C-Type (2)
0|0|0|0| Reserved |LSP Fl=0x02| Reserved | Link Flags
Reserved=0
Length (16) | Class-Num(199) | C-Type (3)


Association Type = 1
Association ID = 168
IPv4 Association Source = Node 1
Global Association Source = 0

13 Appendix V: Summary of Recovery Extensions

The following extensions were made as part of the Recovery Amendment:

- Defined a new OIF_VENDOR_PRIVATE_EXTENSION_TYPE_1 sub-object in Table 5, Table 6, and section 6.3.11.3
- Added support for REPORTING_OSPF_AREA and LINK_EXCLUSIONS TLVs (ERROR_SPEC sub-objects) in section 6.3.1;
- Addressed Recovery compatibility with E-NNI in section 7.3.
- Updated references to OIF documents and added references to IETF RFC4872, RFC4873, RFC6689 and RFC6780 in section 8.
- Added examples for Combined 1+1 protection and soft/hard rerouting in section 11.

14 Appendix VI: List of companies belonging to OIF when document is approved

Acacia Communications
ADVA Optical Networking
Agilent Technologies R & D
Alcatel-Lucent
Altera
AMSS
Amphenol
Anritsu
Applied Communication Sciences
AT&T
Avago
Broadcom
Brocade
Centellax
China Telecom
Ciena
Cisco
ClariPhy
Coriant
Cortina Systems
CPqD
Department of Defense
Deutsche Telekom
Emcore
Ericsson
FCI USA LLC
Fiberhome Technologies Group
Finisar
Fujikura
Fujitsu
Furukawa Electric Japan
Google
Hewlett Packard
Hitachi
Hittite Microwave
Huawei Technologies
IBM
Infinera
Inphi
Intel
JDSU
Juniper Networks
Kaiam

Kandou
KDDI R & R Laboratories
LeCroy
LSI
Luxtera
M/A-COM Technology Solutions
Marben Products
Mellanox
Metaswitch
Mindspeed
Mitsubishi Electric
Molex
MoSys
MultiPhy
NEC
NeoPhotonics
NTT
Oclaro
Optelian
Orange
PETRA
PMC Sierra
QLogic
Ranovus
Semtech
Skorpios
Sumitomo Electric
Sumitomo Osaka Cement
TE Connectivity
Tektronix
Tellabs
TELUS Communications
TeraXion
Texas Instruments
Time Warner Cable
TriQuint Semiconductor
u2t Photonics AG
US Conec
Verizon
Xilinx
Xtera Communications
Yamaichi Electronics